E.T.S.I.S. TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA
PLAN 2000

TEMA: Aplicaciones Electrónicas a la Fórmula SAE

TÍTULO: Pantalla táctil para el piloto en un monoplaza de la Fórmula SAE

AUTOR: Javier Herrero Rubio

TUTOR: Pedro Cobos Arribas **V° B°.**

DEPARTAMENTO: SEC ▾

Miembros del Tribunal Calificador:

PRESIDENTE: Manuel Vázquez López

VOCAL: Pedro Cobos Arribas

VOCAL SECRETARIO: José Antonio Herrera Camacho

DIRECTOR:

Fecha de lectura: septiembre de 2014

Calificación: **El Secretario,**

RESUMEN DEL PROYECTO:

En este proyecto se ha diseñado y fabricado un sistema basado en microcontrolador para un monoplaza que compite en la Fórmula SAE.

El sistema mostrará en una pantalla táctil toda la información que pueda resultar de utilidad al piloto del vehículo. La información mostrada provendrá de los diferentes sensores del monoplaza y será transmitida a través de comunicación CAN Bus.

Así mismo, el sistema permitirá al piloto seleccionar diferentes configuraciones para la conducción a través del control táctil de la pantalla.

RESUMEN

En este proyecto se ha desarrollado un sistema electrónico para un vehículo de Fórmula SAE. La Fórmula SAE es una competición orientada a estudiantes que se basa en el diseño y fabricación de un vehículo de carreras. Este vehículo será posteriormente testeado en una competición a nivel mundial.

El principal objetivo de este proyecto es el estudio, diseño y creación de un sistema para la visualización de información en un vehículo a través de una pantalla táctil. El núcleo del sistema será un microcontrolador de 32 bits de Microchip programado en C sobre un sistema de desarrollo integrado.

El sistema mostrará información que pueda ser de utilidad para el piloto del coche. La información que se mostrará en la pantalla provendrá de los diferentes sensores del propio vehículo (velocidad, rpm, temperatura, estado de la batería). Dichos sensores se comunicarán con el sistema a través de comunicación CAN Bus. Para el testeo del sistema se utilizará una herramienta de simulación CAN.

Además de mostrar información, el piloto será capaz de seleccionar entre diferentes configuraciones para la conducción desde el propio volante. El sistema contiene además los elementos necesarios para la programación y depuración del microcontrolador PIC.

ABSTRACT

In this project, an electronic application for a Formula SAE vehicle has been developed. The Formula SAE is a student-oriented competition based on the design and manufacture of a race car. This car will be later tested in a worldwide competition.

The principal aim of this project is the study, design and manufacture of a system for the display of a vehicle’s information through a touch screen. The system core will be a 32-bit Microchip microcontroller programmed in C code over an Integrated Development Environment.

The system will display useful information to the car driver. The information shown on the screen will come from the different sensors of the vehicle itself (speed, rpm, temperature, battery status). Those sensors will communicate with the system via CAN Bus. A CAN Bus simulator device will be used during the design testing.

In addition to displaying information, the pilot will be able to select different driving configurations from the steering wheel itself. The system also contains the necessary elements for programming and debugging the PIC microcontroller.

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS Y TABLAS.

CAPÍTULO 1. INTRODUCCIÓN

El proyecto desarrollado es una aplicación electrónica para un vehículo tipo fórmula que compite en la **Fórmula SAE**. La Fórmula SAE es una competición de diseño de monoplazas enfocada a estudiantes de diferentes universidades de todo el mundo. Está organizada por SAE Internacional, formalmente conocida como Sociedad de Ingenieros de Automoción. Se entiende.

El objetivo de cada equipo participante en la Fórmula SAE es el diseño y la construcción y la puesta a punto de un monoplaza según unas reglas marcadas por la organización. Posteriormente las diferentes universidades compiten entre sí en diferentes pruebas. En las pruebas estáticas se valorará la viabilidad del proyecto, el cumplimiento de las especificaciones técnicas y el diseño. Por otro lado, en las pruebas dinámicas se pondrá a prueba los elementos propios de un vehículo de competición como la aceleración, el consumo, la estabilidad, o la resistencia entre otros.

El equipo español participante formado en 2003 se conoce como UPMRacing y está formado por estudiantes de diferentes universidades pertenecientes a la Universidad Politécnica de Madrid. Además cuenta con la colaboración del Instituto Universitario de Investigación del Automóvil (INSIA) adscrito a la Escuela Técnica Superior de Ingenieros Industriales e integrado en el Parque Científico y Tecnológico de la UPM.

A lo largo de esta sección se introducirá al lector en los conceptos teóricos más importantes para facilitar la comprensión de los siguientes capítulos en los que se abordará el desarrollo hardware y software del proyecto en profundidad.

1.1. EVOLUCIÓN DE LA VISUALIZACIÓN DE INFORMACIÓN EN LOS VEHÍCULOS DE COMPETICIÓN.

Los elementos de los vehículos de competición cambian año a año a medida que evoluciona la tecnología. Uno de los elementos que han presenta mayores cambios en los últimos años son los volantes de competición. Para hacernos una idea y tomando como ejemplo la Fórmula 1, en 1993 el equipo Sauber contaba con un volante con sólo 2 botones. Hoy en día los volantes utilizados por las diferentes escuderías cuentan con más de 30 botones.

Además de buscar un diseño cada vez más ergonómico y adaptable a las manos de cada piloto, los volantes han evolucionado sensiblemente hacia el mundo digital. Cada vez es más habitual encontrarnos pantallas integradas para mostrar información que pueda ser de utilidad al piloto, ya sean tiempos totales o parciales, revoluciones, etc…

Figura 1.1. Evolución de los volantes en Fórmula 1(1950-1989-2008).

En la actualidad (2014) se opta por modelos híbridos en los vehículos de carreras formados por un conjunto de botones y potenciómetros junto con una pantalla.

Así mismo, en los vehículos de tipo turismo, se está produciendo una introducción paulatina tanto de **pantallas** para visualización de información acompañadas de **paneles táctiles** para facilitar la navegación por los diferentes menús. (ej: radio, visualización velocidad/revoluciones…).

A continuación se expondrá un análisis sobre la visualización de información en un volante de competición desde dos perspectivas diferentes: la visualización mediante pantallas digitales y la interacción piloto-vehículo a través del volante.

1.1.1. Visualización mediante pantallas digitales.

Cuando hablamos de visualización nos referimos a toda aquella información que pueda resultar útil al piloto durante una prueba. Como ejemplo de datos interesantes encontraríamos algunos como la velocidad, la marcha, el tiempo por vuelta, el tiempo al piloto perseguidor etc.

Algunos de estos datos como por ejemplo la marcha actual se llevan mostrando durante muchos años con un sencillo display de 7 segmentos. Sin embargo, cada vez es más necesario que los pilotos manejen mucha **más cantidad de información** durante la conducción. Es en este momento en el que **surge la necesidad** de visualizar más información en unas dimensiones limitadas como son las de un volante. Así, en 2014 se introdujo la primera pantalla LCD en un volante de Fórmula 1.

Figura 1.2. Volante del equipo McLaren 2014.

Además, al tratarse de una **tecnología fácilmente modificable por software**, seremos capaces de modificar la información mostrada en el display de una manera sencilla sin tener que modificar el hardware del mismo. Esto evita gastos innecesarios y limita los costes de fabricación.

1.1.2. Interacción piloto-volante

Tanto o más importante que la visualización de información es la interacción del piloto con el propio vehículo a través del volante. En este caso las ventajas de los botones y potenciómetros utilizados actualmente en competición son claramente superiores a por ejemplo un panel táctil sobre una pantalla LCD. No quiere decir que próximamente no veremos paneles táctiles en los volantes de competición, sino que en este momento la tecnología táctil no está tan avanzada como para su incorporación.

Sin embargo y viendo como ha avanzado la interacción en otros sectores (ejemplo: sector de la telefonía móvil), no sería de extrañar que se introdujeran paulatinamente paneles táctiles en la alta competición.

1.2. PANTALLAS TÁCTILES.

1.2.1. Introducción a las pantallas táctiles.

Una pantalla táctil es un dispositivo electrónico visual que permite al usuario la entrada de órdenes mediante la pulsación directa del panel controlador.

Pese a que comúnmente se habla de **pantalla táctil** (***touchscreen***) como un único elemento, se compone de dos partes claramente diferenciadas: una pantalla (de cualquier tipo) y un **panel táctil** (***touchpanel***) adosado a ella que le confiere la propiedad al conjunto. En los siguientes capítulos de la memoria detallaremos los principios de funcionamiento y los tipos de paneles táctiles más utilizados en la actualidad.

1.2.2. Historia.

La tecnología táctil es relativamente nueva. Fue descrita por primera vez en 1967 por E.A.Johnson en un pequeño artículo y más detalladamente con un ejemplo de aplicación para el control táctil del tráfico aéreo en 1968.

A principios de los años 70, Frank Beck y Bent Stumpe, ingenieros del CERN, desarrollaron un panel táctil transparente que fue posteriormente construido por el CERN y puesto en funcionamiento en 1973 (imagen).

El siguiente gran avance en esta tecnología se produjo simultáneamente en 1971, cuando el Doctor Sam Hurts desarrolló un sensor táctil conocido como "**Elograph**" que fue patentado por la Universidad de Kentucky. En 1973 el Elograph fue reconocido como uno de los 100 nuevos productos tecnológicos más influyentes de la época.

Figura 1.3. Doctor Sam Hurts. Fundador de Elographics.

Posteriormente en 1983, se lanzó al mercado uno de los primeros computadores con pantalla táctil para uso comercial. El HP-150 utilizaba tecnología infrarroja para detectar la posición de cualquier objeto no transparente sobre una pantalla de 9 pulgadas.

En los 90, y con la imparable irrupción de los teléfonos celulares, fueron incrementándose los dispositivos con tecnología táctil. En 1993 se introdujo al mercado la PDA Newton de Apple con reconocimiento de escritura a mano. IBM también entró en el sector táctil con IBM Simon en 1994, el que convertiría en el primer teléfono inteligente del mercado.

Figura 1.4. PDA Newton. IBM Simon.

La popularidad y el uso de la tecnología táctil se fue incrementando exponencialmente desde entonces. En el 2004 Nintendo lanzó la videoconsola portátil Nintendo DS que contaba con una pantalla táctil y que se ha convertido, junto a posteriores versiones de la misma, en la segunda consola portátil más vendida con más de 150 millones de unidades vendidas.

En 2007 y con Steve Jobs a la cabeza, Apple lanzó el celular iPhone que contaba solamente con una pantalla y la tecnología táctil suponiendo una auténtica revolución en el mercado de los teléfonos celulares. En 2014 Apple anunció que las cifras de iPhone habían alcanzado los 500 millones de unidades vendidas.

Hoy en día el uso de la tecnología táctil está muy extendido como medio de entrada-salida sustituyendo, en muchos casos, a otras tecnologías anteriores. Debido, entre otros factores, a sus bajos costes de fabricación, podemos encontrar paneles táctiles en maquinaria industrial, sistemas domóticos, máquinas expendedoras, electrodomésticos, y un largo etcétera.

Figura 1.5. Pantalla táctil en electrodoméstico Samsung.

1.2.3. Tipos de paneles táctiles.

La rápida evolución tecnológica ha originado la aparición de multitud de paneles táctiles según la tecnología utilizada para la detección de la pulsación. Actualmente, los paneles resistivos y capacitivos son los más extendidos.

A continuación detallaremos y los compararemos entre sí. Por último, comentaremos otros tipos de tecnologías menos comunes como el SAW (sistema de onda acústica de superficie) o la tecnología infrarroja.

1.2.3.1. Paneles táctiles resistivos.

Los paneles resistivos se componen de varias capas superpuestas entre las que existe una separación de un material conductor.

Una vez que se realiza una pulsación las dos caras internas conductoras se ponen en contacto. En ese punto se realiza la medición de voltaje. Para ello se aplica voltaje en una capa generando un gradiente en toda la superficie. A continuación se procede a la medición de la tensión correspondiente en la otra capa. Se seguirá el mismo proceso para la otra coordenada. De esta forma una de las capas se utiliza para obtener la posición en el eje X y la otra la posición en el eje Y.

Una vez obtenidas ambas medidas y mediante un conversor analógico digital (ADC) podremos convertir las tensiones medidas a sus coordenadas x e y equivalentes en nuestra pantalla. Este tipo de panel se conoce también como panel **resistivo de 4 hilos**.

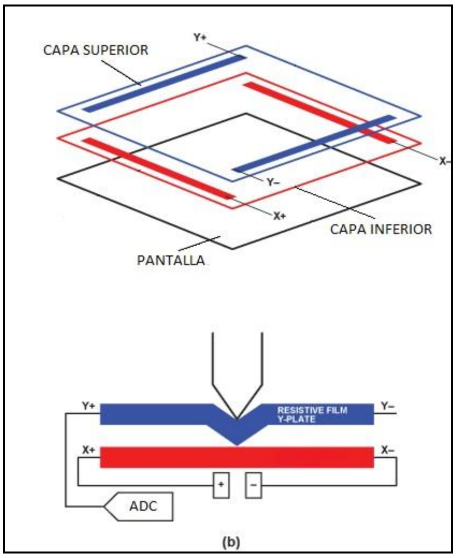


Figura1.6. Funcionamiento de un panel resistivo.

Cabe mencionar dos variaciones en tecnología de paneles resistiva al panel de 4 hilos que añaden algunas mejoras adicionales. El panel **resistivo de 8 hilos** evita la recalibración del panel cada cierto tiempo; por otro lado, el panel **resistivo de 5 hilos** añade durabilidad y mayor resistencia contra agentes externos tales como agua o polvo.

1.2.3.2. Paneles táctiles capacitivos.

El panel capacitivo consiste en una capa cubierta con un material de óxido de indio y estaño que conduce una corriente eléctrica a través de un sensor.

Mediante la aplicación de un voltaje en las 4 esquinas del sensor capacitivo, se crea un campo uniforme. Al realizar una pulsación con un dispositivo conductivo se provoca un cambio de corriente en cada lado en proporción a la distancia a cada esquina. Es entonces cuando la controladora calcula la posición del dedo a partir del cambio de flujo.

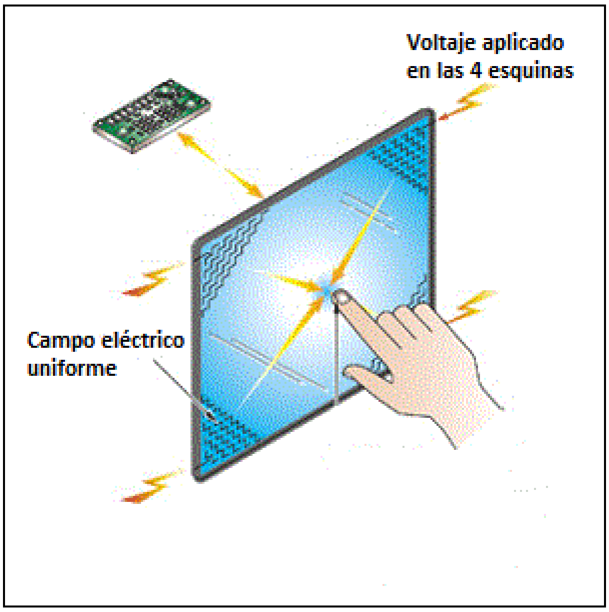


Figura1.7. Funcionamiento de un panel capacitivo.

1.2.3.3. Otras tecnologías.

Aunque los paneles capacitivos y resistivos son las más utilizadas en procesamiento táctil existen otras alternativas tecnológicas a destacar.

La tecnología conocida como **SAW(Surface Acoustic Wave)** que utiliza las ondas de ultrasonido que se transmiten por la pantalla para captar la posición de la pulsación.

Otra tecnología es la **infrarroja** que basa su funcionamiento en la generación de una matriz infrarroja a lo largo de la pantalla de manera que al realizarse una pulsación el haz infrarrojo es interrumpido en ambos ejes obteniéndose así la coordenada deseada.

1.2.3.4. Ventajas y desventajas de los paneles resistivos sobre los capacitivos.

Ventajas:

- Más económicos.
- El polvo y el agua no afectan a su funcionamiento.
- Funcionan con cualquier objeto capaz de aplicar presión. Las capacitivas en cambio necesitan utilizarse con los dedos o con lápices o guantes especiales.
- Mayor rango de temperatura de funcionamiento.

Desventajas:

- Menor visibilidad debido al número de capas.
- Poca durabilidad. Muy vulnerables a golpes y arañazos.

1.3. MICROCONTROLADOR.

Un microcontrolador es un circuito electrónico programable capaz de ejecutar las órdenes almacenadas en su memoria. Se compone de varios grupos funcionales: una unidad central de procesamiento (CPU), unidades de memoria, dispositivos de entrada y salida y periféricos. Todo ello en un mismo circuito integrado o chip gracias a su alta densidad de integración.

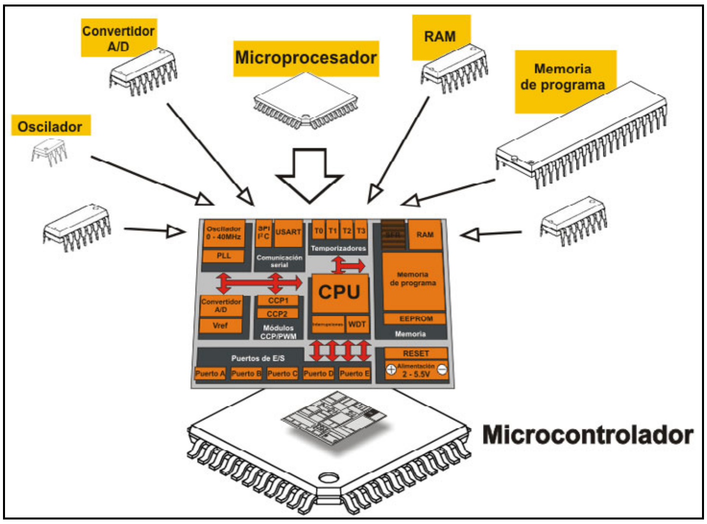


Figura1.7. Estructura de un microcontrolador.

1.3.1. Microcontroladores PIC.

Se conoce como PIC (Peripheral Interface Controller) a los microcontroladores fabricados por Microchip Technology Inc. Microchip es una empresa fabricante de microcontroladores, memorias y semiconductores analógicos.

Los microcontroladores PIC se basan en la **arquitectura Harvard** mediante la cual la CPU está conectada a dos memorias por medio de dos buses separados. La Memoria de Programa contiene las instrucciones del programa y la Memoria de Datos sólo almacena los datos. Esta independencia en las memorias permite diseñar buses con diferente ancho y permite que la CPU acceda a datos mientras ejecuta la siguiente instrucción logrando una mayor velocidad de operación.

La arquitectura de CPU es de tipo **RISC** componiéndose de un número reducido de instrucciones con el objetivo de posibilitar la segmentación y el paralelismo en la ejecución de instrucciones y reducir los accesos a memoria.

Por otro lado, se trata de un microcontrolador **muy extendido**, por lo que existe mucha información sobre ellos ya sea en libros, manuales, revistas especializadas, internet, etc. Por su parte, Microchip pone a disposición del usuario una gran cantidad de información para facilitar el manejo de sus dispositivos. También ofrece **software gratuito básico** para la programación, así como la posibilidad de obtener muestras gratuitas de algunos de sus componentes.

Podemos clasificar los PICs según el tamaño de datos que manejan. Pueden ser de 4, 8, 16 y 32 bits, siendo estos últimos los utilizados en este proyecto y los que mayor crecimiento están experimentando en los últimos años.

1.3.2. Familia de microcontroladores PIC de 32 bits.

La familia de microcontroladores de 32 bits se introdujo en 2007 y actualmente son los procesadores de alta gama de Microchip.

Entre los microcontroladores de 32 bits existe una gran diversidad en el mercado, lo que facilita la elección según las necesidades propias de cada usuario. La familia de propósito general de 32 bits se conoce como PIC32MXxxx .

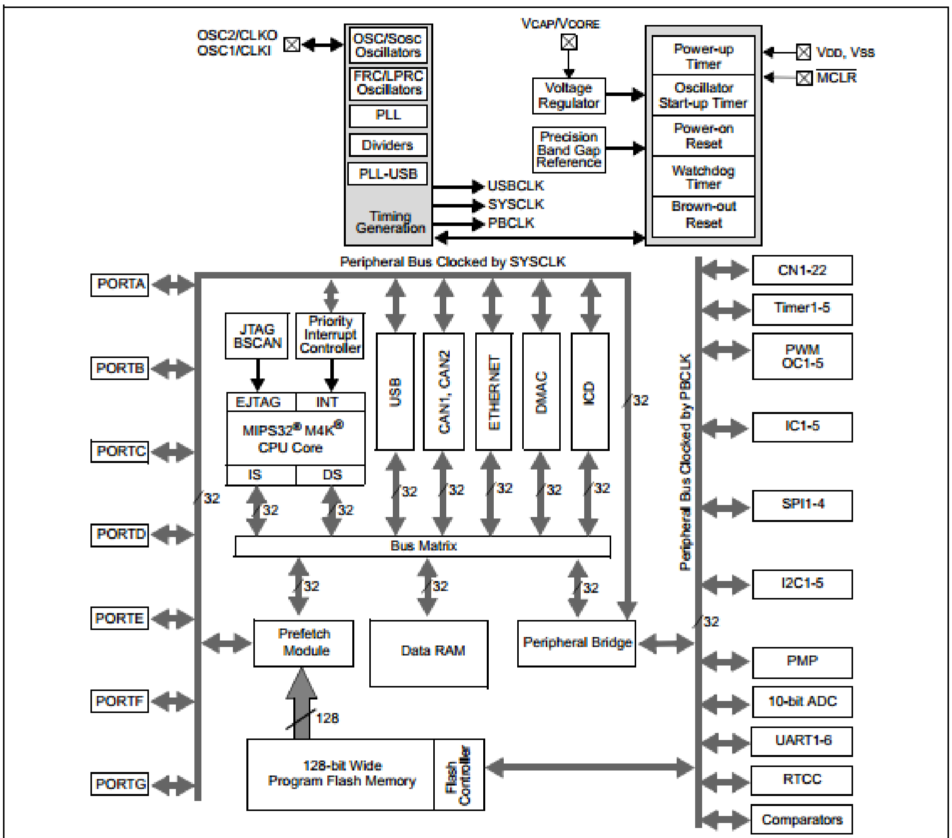
Las características principales del PIC32MXxxx son las siguientes:

- Procesador de 32 bits MIPS32® M4K® con arquitectura RISC con una frecuencia de trabajo máxima de 80MHz.
- Voltaje de alimentación de 2.3 a 3.6V.
- Memoria Flash de 64 a 512KB.
- Memoria SRAM de 16 a 128KB.
- Compatibilidad con la mayoría de dispositivos PIC24/dsPIC®.
- Múltiples modos de ahorro de energía.
- Múltiples vectores de interrupción con prioridad individual programable.
- Modo de reloj a prueba de fallos.
- Watchdog configurable.

El PIC32 cuenta además con un número elevado de periféricos:

- Hasta 8 canales hardware DMA (Acceso Directo a Memoria).
- USB 2.0 con controlador OTG y canal DMA dedicado.
- 10/100 Mbps Ethernet MAC con interfaz MI, RMII y canal DMA dedicado.
- Módulo CAN 2.0B con soporte para direccionamiento DeviceNet y canal DMA dedicado.
- Capacidad para oscilador externo de 3MHz hasta 25MHz.
- Oscilador interno de 8 y 32MHz.
- 6 módulos UART con soporte para RS-232, RS-485 y LIN.
- Hasta 4 módulos SPI.
- Hasta 5 módulos I^2C.
- PLLs separados para relojes de CPU y USB.
- Puerto Paralelo Maestro y Esclavo (PMP/PSP) con datos de 8 y 16 bits, y 16 líneas de direccionamiento.
- Reloj y calendario en tiempo real (RTCC).
- 5 contadores/timers de 16 bits.
- 5 entradas de captura.

- 5 comparadores/PWM de salida.
- 5 pines de interrupción externa.
- Múltiples pines de entrada-salida de alta velocidad.
- 16 canales ADC de 10 bits.
- 2 comparadores analógicos.



Note 1: Some features are not available on all device variants.
2: BOR functionality is provided when the on-board voltage regulator is enabled.

Figura1.9. Diagrama de bloques. PIC32.

1.4. CAN BUS

1.4.1. Introducción

El bus CAN (Controller Area Network) es un bus de datos de comunicación serie empleado en sistemas distribuidos en tiempo real. Pese a que originalmente fue desarrollado para aplicaciones en la industria automotriz, debido a sus características, robustez y excelente relación calidad/precio, fue adoptado para aplicaciones industriales y de control.

El sistema CAN fue originalmente desarrollado y especificado en los años 80 por la compañía alemana Robert Bosch en un intento por resolver los problemas de comunicación entre diferentes sistemas de control de los vehículos. La primera especificación del CAN se mejoró y amplió con las especificaciones 2.0. que presentó Bosch en 1991. A partir de 1992 se comenzó a utilizar masivamente en la industria automovilística. Aunque diferentes empresas crearon sus especificaciones CAN, en 1993 se publicó un estándar ISO 11898 que fue ampliado posteriormente con la aparición del protocolo unificado de la CiA (Can in Automotion) para el CAN, el CANopen.

Pese a que la industria automovilística representa el mayor porcentaje de uso del bus CAN, debido a sus características, un gran número de sectores se aprovechan de sus ventajas.

1.4.2. Características.

-Económico y sencillo: son dos de las razones que motivaron su desarrollo por la necesidad de economizar el coste y de minimizar la complejidad de cableado, algo muy importante en el sector automovilístico debido a las restricciones de espacio. A pesar de que el medio de transmisión sea un par trenzado de cables, el sistema CAN también es capaz de trabajar con un solo cable si fuese necesario.

-Orientado a mensajes y no a direcciones: La información que se va a transmitir se descompone en mensajes. Cada mensaje tiene un identificador único dentro de la red. El mensaje por tanto no va direccionado a ninguna unidad en concreto, sino que cada una de ellas reconocerá mediante el identificador si le interesa el mensaje o no.

-Estructura de campos definida: La información que circula a través del bus son paquetes de bits con una longitud limitada y una estructura definida de campos.

-Alto número de nodos: La posibilidad de conectar un gran número de nodos es fundamental en sistemas integrados en un vehículo como en el caso del presente proyecto.

-Desconexión de nodos defectuosos: Si un nodo falla en algún momento el sistema es capaz de identificarlo y aislarlo permitiendo así que no afecte al sistema completo.

-Velocidad flexible: según las necesidades del sistema a desarrollar las redes CAN están definidas como red de alta velocidad (hasta 1Mbps) o de baja velocidad tolerante a fallos (menor o igual a 125 Kbps). La velocidad dependerá también de la distancia a la que necesitemos transmitir.

-Sistema multimaestro: Cuando el bus está libre, cualquier nodo puede empezar la transmisión de un mensaje. El mensaje con mayor prioridad ganará la arbitración del bus.

-Detección de errores: CAN posee una gran capacidad de detección de errores. El sistema cuenta con una serie de mecanismos que aseguran que el mensaje es transmitido y recibido correctamente. En otro caso, el sistema es capaz de retransmitir automáticamente las tramas erróneas.

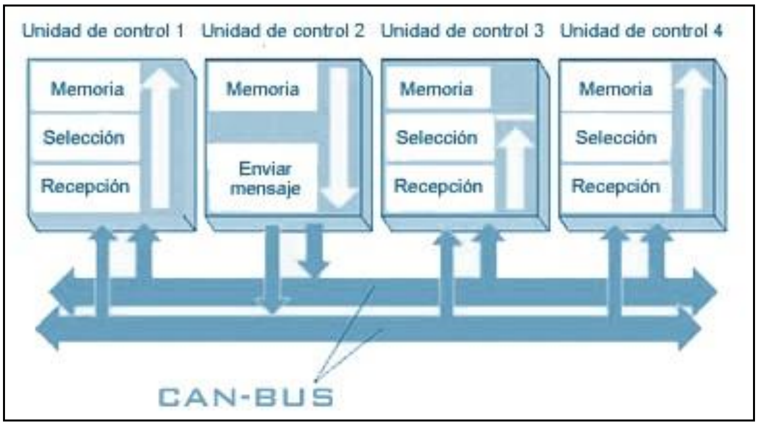


Figura1.10. Estructura CAN Bus.

1.4.3. El modelo OSI. Capa física y capa de enlace.

El protocolo CAN está estructurado de acuerdo con el modelo OSI, formado por siete capas que define las diferentes fases por las que deben pasar los datos para viajar de un dispositivo a otro sobre una red de comunicaciones. CAN define dos capas el modelo, la capa física y la capa de enlace de datos.

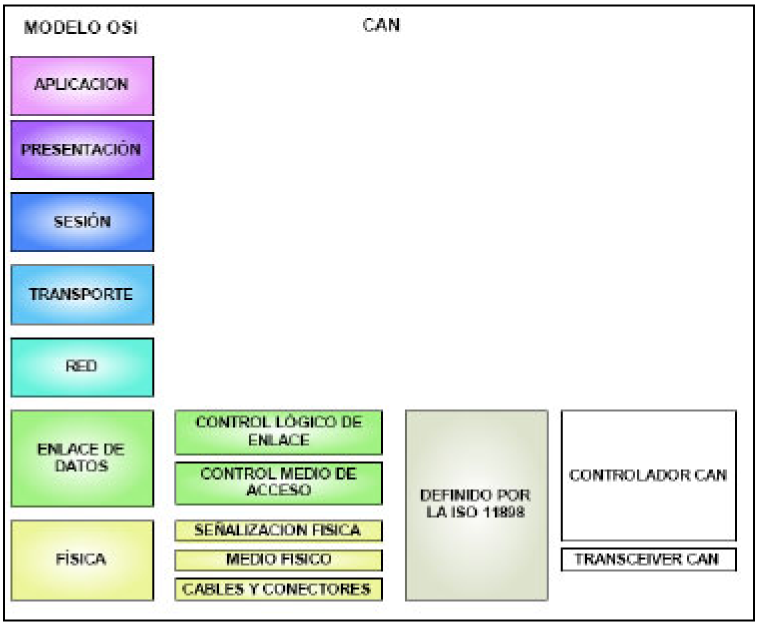


Figura1.11. Modelo OSI en el CAN Bus.

1.4.3.1. Capa física.

La capa física define parámetros tales como niveles de señal, codificación, sincronización y tiempos en que los bits transfieran al bus.

La información circula por dos cables trenzados unidos a todas las unidades de control que forman el sistema completo. Toda la información se transmite por diferencia de tensión entre los dos cables. Los niveles de tensión del CAN son los siguientes:

- **Dominante** (‘0’ lógico): la tensión diferencial (CAN_H – CAN_L) es del orden de 2V con CANH = 3.5V y CANL = 1.5V.
- **Recesivo** (‘1’ lógico): la tensión diferencial es del orden de 0V con CAN_H = CAN_L = 2.5V.

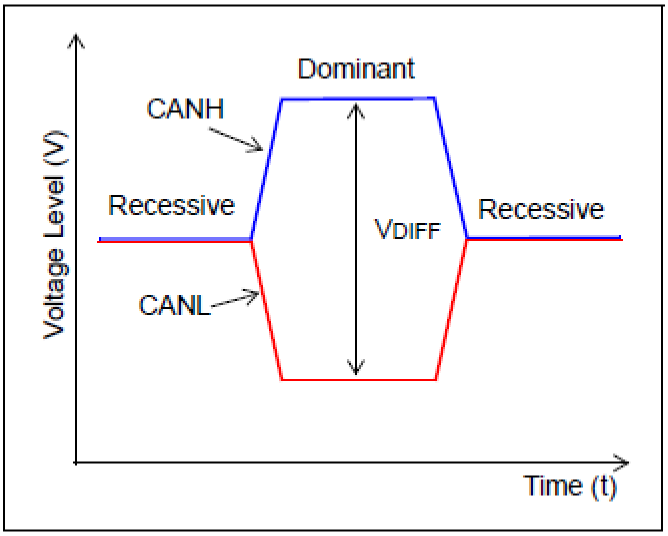


Figura1.12. Niveles de tensión CAN.

1.4.3.2. Capa de enlace.

La capa de enlace se divide en dos: el *subnivel LLC* (Control Lógico de Enlace) y el *subnivel MAC* (Control de Acceso al Medio).

El subnivel LLC maneja las notificaciones, el filtrado de mensajes y el restablecimiento de las funciones de administración.

Por su parte, el subnivel MAC se encarga del encapsulamiento y desencapsulado de datos, de la detección y control de errores, el proceso de bit stuffing y la transmisión serie. El protocolo de comunicación CAN utiliza un control de acceso al medio de tipo "CSMA/CD+AMP" (Acceso Múltiple con Detección de Portadora, Detección de Colisiones y Arbitraje con Prioridad de Mensaje). De este modo los nodos que deseen transmitir información deberán esperar a que el bus esté libre (Detección de Portadora); una vez se cumpla esta condición los nodos transmiten un bit de inicio (Acceso Múltiple). Cada nodo lee el bus bit a bit durante la transmisión de la trama y comparan el valor que se transmite con el valor recibido. Si se detecta una diferencia en los valores de los bits se lleva a cabo el mecanismo de arbitraje mediante prioridad.

1.4.4. Máscaras y filtros.

Debido a que CAN es un protocolo que se basa en tipos de mensajes y no en direcciones, los mensajes no son enviados de un nodo origen a uno destino. Cualquier mensaje es recibido por todos los nodos que se conecten al bus. Los nodos por tanto deben programar un test de aceptación para determinar que mensajes procesan y cuáles no. El campo de la trama testeado es el identificador. Este filtrado de mensajes se realiza mediante el uso de filtros y máscaras.

1.4.5. Tipos de tramas.

CAN utiliza mensajes con una estructura predefinida también conocidos como tramas. El protocolo CAN define cuatro tipos principales de tramas para la gestión de la transferencia de mensajes. Las tramas de error y sobrecarga se encargan de la gestión y manipulación de errores; la trama de datos se utiliza para poner información en el bus; la trama de información remota es enviada por un nodo para solicitar una transmisión de una trama de datos a otro bus.

1.4.5.1. Trama de datos.

En un bus CAN los nodos transmiten información espontáneamente con tramas de datos, bien sea por un proceso cíclico programado o bien activado mediante eventos en el nodo.

La trama de datos consta de una serie de campos de diferente tamaño. Son los siguientes campos:

- *Inicio de trama (SOF)*: Es una celda de un único bit dominante que indica el inicio de la trama de datos y sirve para la sincronización con otros nodos.
- *Campo de arbitraje*: Está formado por el identificador y el bit RTR que será dominante en una trama de datos y recesivo en el caso de una trama remota. Según sea el formato de la trama estándar (CAN 2.0A) o extendido (CAN 2.0B), el identificador tendrá 11 ó 29 bits.
- *Campo de control*: el primero de los 4 bits (IDE) indica si la trama es de CAN estándar o extendido; el bit r0 es siempre recesivo; los dos bits que forman DLC informan del número de bytes (de 0 a 8) que componen el campo de datos.
- *Campo de datos*: contiene el mensaje a transmitir y puede contener entre 0 y 8 bytes.
- *Código de redundancia cíclica (CRC)*: es un campo utilizado para comprobar el correcto envío de la trama produciéndose error de CRC en caso contrario.
- *Celda de reconocimiento(ACK)*: Es un campo de 2 bits que indica la correcta recepción del mensaje. El nodo que transmite el mensaje manda este bit como recesivo y cualquier nodo que lo reciba lo pondrá a dominante para indicar que se ha recibido el mensaje.
- *Fin de trama(EOF)*: 7 bits recesivos que indican el final de la trama.
- *Espaciado entre tramas (IFS)*: Mínimo de 3 bits recesivos para separar varias tramas CAN seguidas.

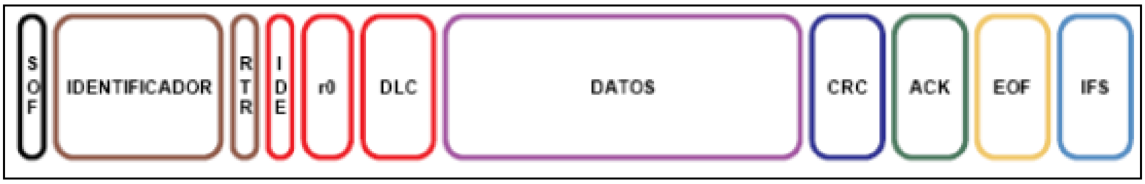


Figura1.13. Trama de datos CAN.

1.4.5.2. Trama de información remota.

Con la trama de información remota un nodo es capaz de enviar una petición de trama de datos a otro nodo. El formato de trama es el mismo que la trama de datos con la diferencia de que el campo RTR será recesivo indicando el tipo de trama remota y no contendrá ningún dato. El identificador en este caso, corresponderá al mensaje que se solicita y podrán ser 11 ó 29 bits dependiendo si es una trama remota estándar o extendida.

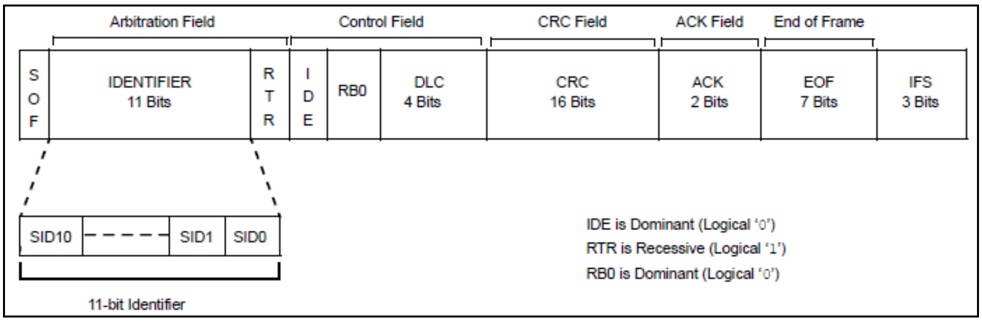


Figura1.14. Trama de información remota CAN.

1.4.5.3. Trama de error.

La trama de error será generada por cualquier nodo que detecte un error. Este tipo de trama se compone de dos únicos campos: Flag de Error y el Delimitador de Error.

El Delimitador de Error consiste en ocho bits recesivos y permite que el nodo reinicie una comunicación limpia justo después de que el error haya ocurrido.

Dependiendo del estado de error que tenga el nodo que lo detecta el Flag de error se comporta de diferente manera. Si un nodo en estado de error “Activo” detecta un error en el bus se interrumpe la comunicación del mensaje en proceso generando un “Indicador de error activo” que consiste en una secuencia de 6 bits dominantes. Esto provocará tramas de error en otros nodos al romper la regla de relleno de bits. Por tanto el indicador de error se puede extender entre 6 y 12 bits dominantes sucesivos. Después se reiniciará la comunicación y el nodo interrumpido reintentará la retransmisión del mensaje.

Si por el contrario un nodo en estado de error “Pasivo” detecta un error, el nodo transmitirá un “Indicador de error pasivo” que consiste en 6 bits recesivos que no afectará a ningún otro nodo de la red como ocurría con el estado “Activo”.

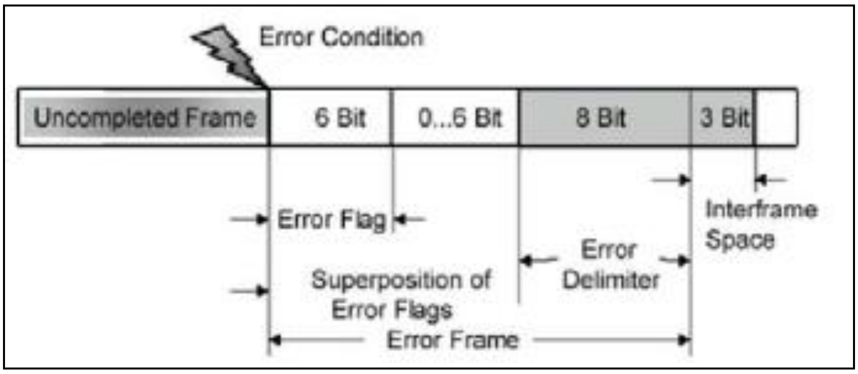


Figura1.15. Trama de error CAN.

1.4.5.4. Trama de Sobrecarga.

Una Trama de Sobrecarga puede ser generada por cualquier nodo que detecte un bit dominante durante el Espaciado Entre Tramas (IFS) o cuando el nodo no está preparado para recibir el siguiente mensaje. Esta trama tiene el mismo formato que una trama de error con la diferencia que solo puede ser generada durante el Espaciado Entre Tramas.

1.4.6. Componentes físicos de una red CAN

Una red que implementa CAN está formada por los siguientes elementos:

- *Cableado*: Está constituido por dos hilos paralelos, trenzados y/o blindados, según requerimientos electromagnéticos. Los segmentos de cable para la conexión de nodos deben ser tan cortos como sea posible, especialmente en tasas altas de bit. Los hilos son conocidos por CAN_H y CAN_L como se indicó en capítulos anteriores.
- *Terminadores*: A cada extremo de la red se conecta una resistencia terminal para cerrar el bus. Permiten adecuar el funcionamiento del sistema ya que impiden fenómenos de reflexión que pueden provocar que no se puedan alcanzar las pendientes necesarias en los flancos de la señal.
- *Nodos*: Cada nodo está individualmente conectado con el bus CAN, pudiendo tanto transmitir mensajes al bus como supervisar todos los mensajes que circulen por el bus.

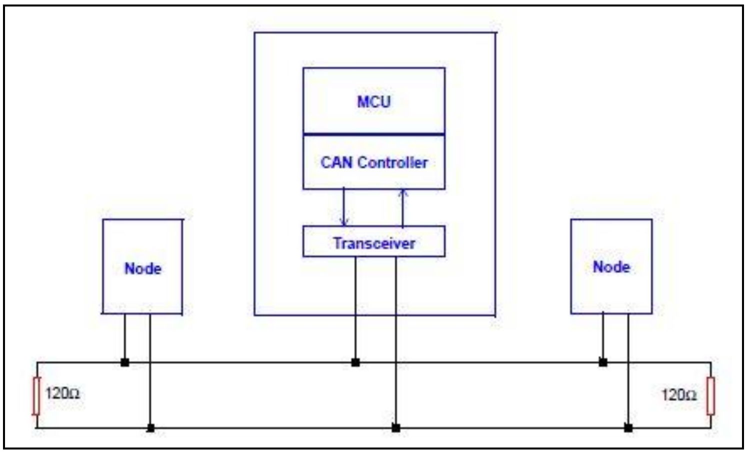


Figura1.16. Componentes de una red CAN.

1.4.7. Estructura de un nodo CAN.

Para la implementación de un nodo CAN son necesarios los siguientes elementos:

- *Microcontrolador*: Es la unidad de control que realizará diferentes funciones según la aplicación requerida. Dará “significado” a los mensajes recibidos por el bus y será el encargado de solicitar un envío de una trama al mismo.
- *Controlador CAN*: Es el encargado de la comunicación entre el microprocesador y el transceptor. Recibe del microprocesador los datos que han de ser transmitidos. Se ocupa del acondicionamiento de la señal y del correcto funcionamiento del protocolo, así como de la detección de errores. Como veremos en el capítulo de hardware de este proyecto, algunos microcontroladores PIC llevan el controlador CAN integrado en los mismos.
- *Transceiver*: El transceptor o transceiver es el elemento que une el controlador directamente a las líneas CAN_H y CAN_L del bus. Transforma los datos que recibe del controlador CAN en señales eléctricas y los transmite. De la misma manera

recibe señales eléctricas del bus y las transforma para el controlador.

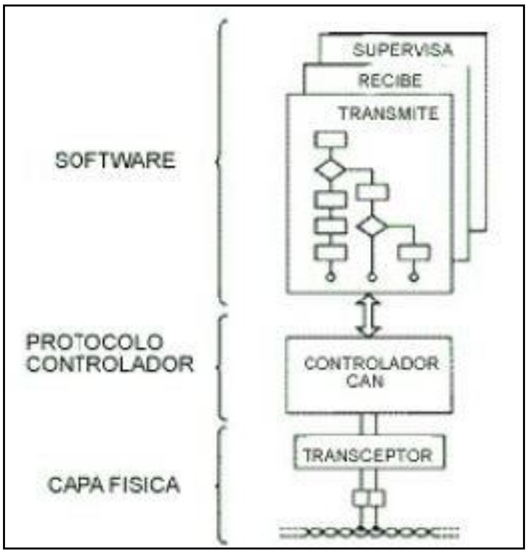


Figura1.17. Estructura de un nodo CAN.

2. DESARROLLO HARDWARE.

Como en todo proyecto de cierta complejidad, se ha abordado el desarrollo hardware en bloques o partes claramente diferenciados. Cada uno de los bloques que se detallarán a continuación tiene su propia función dentro del sistema general.

En este capítulo se comenzará presentando un diagrama de bloques general del sistema completo. A continuación se abordará la elección de los componentes que forman el corazón del sistema (microcontrolador, controlador gráfico y pantalla) y por último se analizarán y justificarán en detalle cada uno de los bloques funcionales del sistema.

2.1. DIAGRAMA DE BLOQUES GENERAL

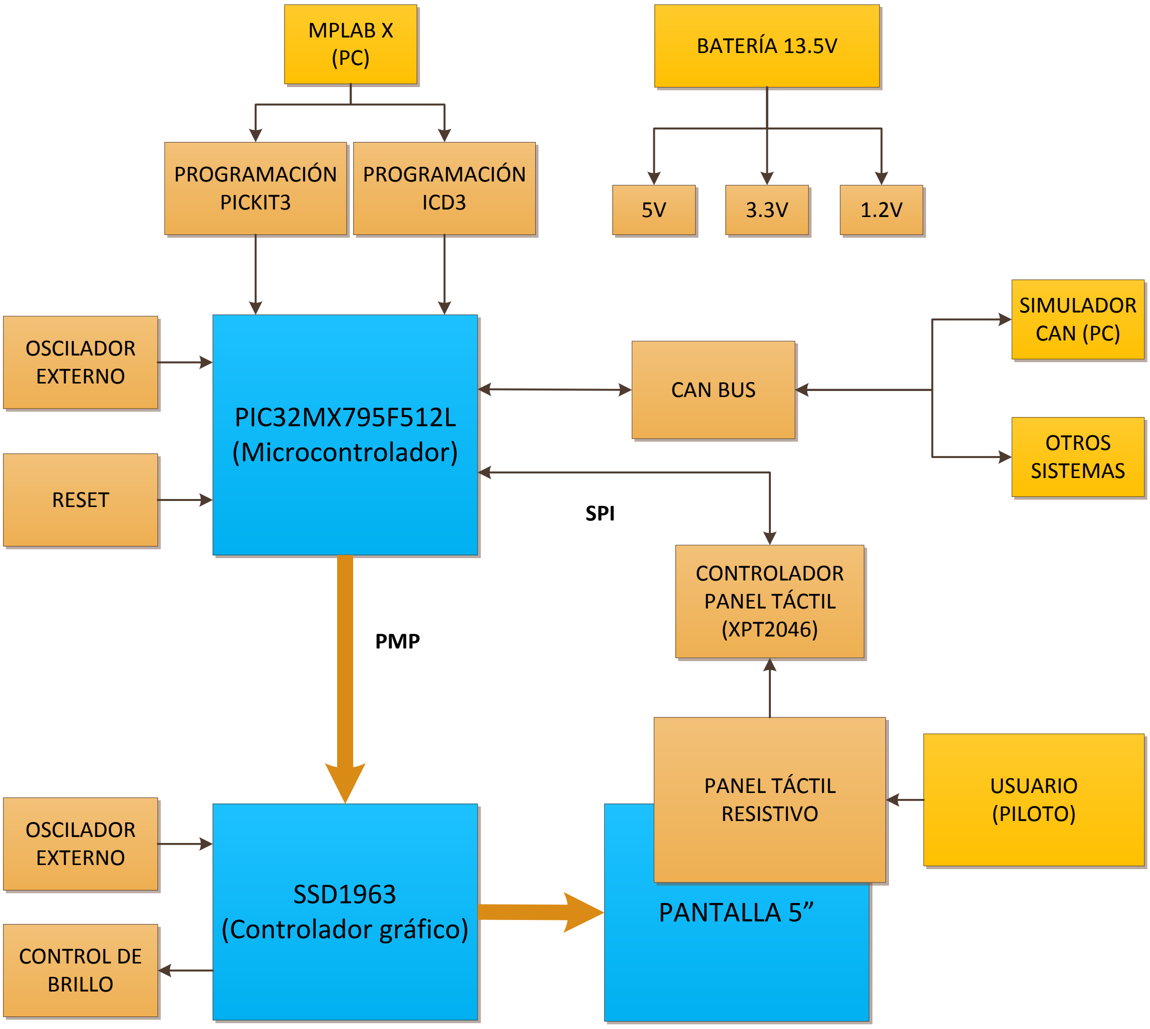


Figura 2.1. Diagrama de bloques general.

2.2. ELECCIÓN DE COMPONENTES.

2.2.1. MICROCONTROLADOR.

Uno de los **requisitos previos** del proyecto era la utilización de un microcontrolador de 32 bits de la familia PIC, fabricados por Microchip Technology Inc. Una de las razones es la posibilidad de aprovechamiento de materiales ya adquiridos para proyectos previos del fabricante Microchip, como por ejemplo el simulador CAN.

Al tratarse de un sistema diseñado para montarse en un único coche y dado que no es necesaria una producción en serie del mismo, se optó por elegir la gama más alta dentro de los microcontroladores de PIC32 (PIC32MX795F512L) . Los factores que llevaron a esa decisión son los siguientes:

- ✓ Es el microcontrolador con mayor memoria de programa (512KB) y de datos (128KB) cumpliendo las necesidades de tamaño del sistema a desarrollar y con posibilidad de aumentar el tamaño del programa en futuros proyectos sin necesidad de cambiar el microcontrolador o de añadir una memoria externa.
- ✓ Dispone de Puerto Paralelo Maestro PMP necesario para la comunicación con el controlador gráfico.
- ✓ Controlador CAN integrado. Las gamas más bajas de 32 bits de PIC no incluyen el controlador CAN, lo que implicaría añadir un integrado más al esquemático. Las gamas media y alta disponen de 1 ó 2 módulos CAN. En el caso del PIC32MX795F512L dispone de 2 módulos del que utilizaremos sólo 1.
- ✓ Posibilidad de pedir muestras gratuitas a Microchip de cualquier microcontrolador de la serie PIC32MX y por tanto un modelo superior no supone un coste extra al presupuesto.

2.2.2. MÓDULO PANTALLA Y PANEL TÁCTIL.

Otro de los requisitos del proyecto, como se comentó en capítulos anteriores, era la búsqueda de una pantalla y un panel táctil crear un sistema completo funcional. Además se debía crear un prototipo de PCB con los componentes necesarios para el funcionamiento del mismo. En un primer momento se acordó que la fabricación de la PCB completa se realizaría mediante un proceso de fabricación semiautomático mediante una máquina de prototipado de PCBs. Con este sistema se facilitaría la creación de las conexiones más críticas como las del microcontrolador (100 pines en formato TQFP) y el controlador gráfico (128 pines en encapsulado LQFP).

Debido a problemas con la herramienta de prototipado ajenos al alumno, se decidió que el proceso de fabricación sería realizado manualmente con ácidos en el laboratorio del departamento. El proceso mediante ácidos es un proceso laborioso pero muy eficiente en circuitos con pistas anchas y pocos componentes, lo cual no era el caso. Esto complicó sobremanera la creación del circuito impreso debido a la escasa distancia entre pistas tanto del microcontrolador como del controlador gráfico. Tras realizar algunas pruebas previas de insolado de PCBs se comprobó que a duras penas se conseguían obtener unas pistas adecuadas para realizar una soldadura manual segura con el microcontrolador. Recalcar que el ancho de pad del PIC32 es de 0.2mm mientras que en el controlador es todavía menor, de 0.18mm. Todo esto se añadió a la dificultad de rutear una PCBs a **doble cara** con tantísimas conexiones a componentes tan pequeños.

El ruteo y la soldadura del PIC fueron extremadamente laboriosos (comentar que 48 de los 100 pines del PIC tenían que ser ruteados). En cambio, el ruteo, insolado y posterior soldadura del controlador gráfico eran inasumibles mediante un proceso manual a doble cara, tanto por el ancho de pines como por el ruteado necesario. El controlador requiere la conexión de más del 90% de los 120 pines, teniendo en cuenta que tiene que conectarse con el PIC, la pantalla, masa, y dos alimentaciones (1.2V y 3.3V).

En este momento se planteaban dos alternativas:

- La construcción de la PCB de forma profesional multicapa. Esta opción fue descartada por falta de presupuesto del departamento.
- Tras realizar una búsqueda exhaustiva en el mercado se obtuvo la segunda solución que resultaría la elegida. Se observó que en el mercado se vendían conjuntos completos que incluían la pantalla, el panel táctil y el controlador gráfico de 120 pines soldado en una PCB multicapa debajo de la propia pantalla, ocupando muy poco espacio. Esta tarjeta cuenta además 40 pines de conexión para conectar un microcontrolador. Sorprendentemente el precio no era elevado teniendo en cuenta todos los componentes que incluía por lo que se tomó la decisión de adquirirlo. El contenido del producto es el siguiente:

  - ✓ Pantalla TFT LCD de 5 pulgadas con una resolución de 800x480 píxeles.

  - ✓ Retroiluminación LED de la pantalla y circuito que permite la configuración del brillo mediante PWM.

  - ✓ Controlador gráfico SSD1963.

  - ✓ Panel táctil resistivo 4 hilos y circuito controlador para panel táctil resistivo de 4 hilos XPT2046.

  - ✓ Circuito regulador lineal de 3.3V a 1.2V necesarios para la alimentación del controlador gráfico.

  - ✓ 40 pines de entrada-salida para la conexión a un sistema externo.

✓ Además también incluye un circuito para la incorporación y manejo de una tarjeta SD (no utilizado en este proyecto).

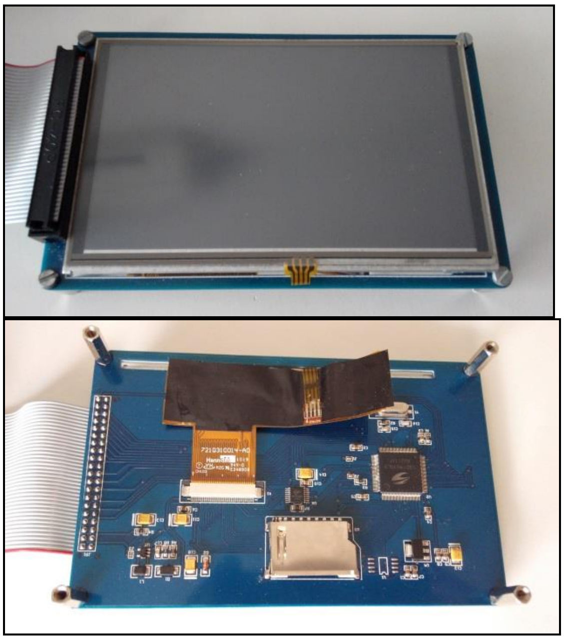
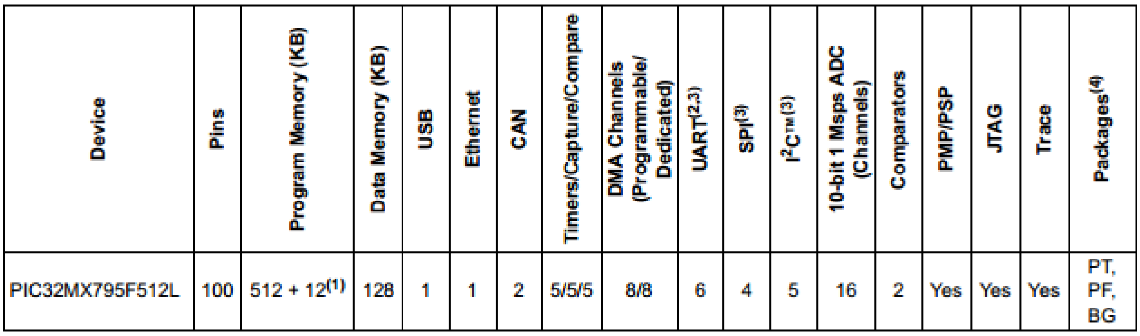


Figura 2.2. Módulo pantalla y panel táctil.

2.3. PIC32MX795F512L

2.3.1. Características.

Las características generales de la familia PIC32MXxxx se pueden encontrar en el capítulo de introducción dedicado a los microcontroladores. Al tratarse del modelo de gama más alta el microcontrolador cuenta con el mayor número de periféricos así como la mayor cantidad de memoria de programa y de datos.



Device	Pins	Program Memory (KB)	Data Memory (KB)	USB	Ethernet	CAN	Timers/Capture/Compare	DMA Channels (Programmable/ Dedicated)	UART[2,3]	SPI[3]	I^2C™[3]	10-bit 1 Msps ADC (Channels)	Comparators	PMP/PSP	JTAG	Trace	Packages[4]
PIC32MX795F512L	100	512 + 12[1]	128	1	1	2	5/5/5	8/8	6	4	5	16	2	Yes	Yes	Yes	PT, PF, BG

Figura 2.3. Características del PIC32MX795F512L.

2.3.2. Conexiones mínimas necesarias.

En el propio datasheet proporcionado por Microchip se nos especifican unas conexiones mínimas recomendadas para el correcto funcionamiento del dispositivo:

- Todos los pines referentes a tensiones tanto de tensión de alimentación del dispositivo (V_{DD}) como la referencia de tierra (V_{SS}) deberán estar conectados a **condensadores de desacoplo**. El valor de los mismos será de 100nF/10-20V. Se recomienda además la utilización de condensadores cerámicos.

- V_{USB} se conectará a V_{DD} en el caso de no ser utilizado.
- La alimentación de módulos analógicos AV_{DD} y AV_{DD} se conectará a un condensador de desacoplo aunque el Convertidor Analógico-Digital (ADC) no se utilice.
- Será necesario un condensador low-ESR (1Ω)/6V en el Regulador de Voltaje Interno (V_{CAP} / V_{CORE}) para estabilizar la salida del regulador de voltaje. El condensador será cerámico o de tántalo.

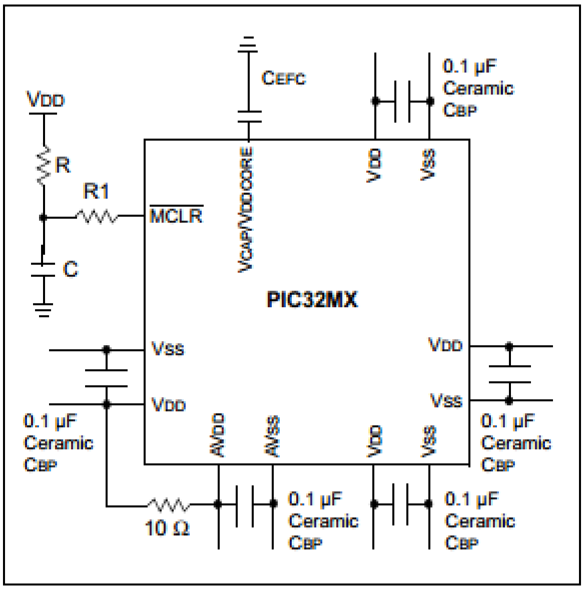


Figura 2.4. Conexiones mínimas del PIC32MX795F512L.

2.3.3. Programación ICSP.

ICSP es el acrónimo de "In Circuit Serial Programming" o "Programación Serie en Circuito". Mediante ICSP podemos grabar la memoria de programa, la memoria de datos y los registros de configuración directamente en la placa. De esta forma, y sin necesidad de retirar el microcontrolador de la tarjeta del circuito seremos capaces de programar o actualizar el software del sistema.

Para la programación del dispositivo se requiere de un programador externo. En nuestro caso, el propio departamento se encargó de proporcionarnos dos herramientas de programación para los PIC de 32 bits.

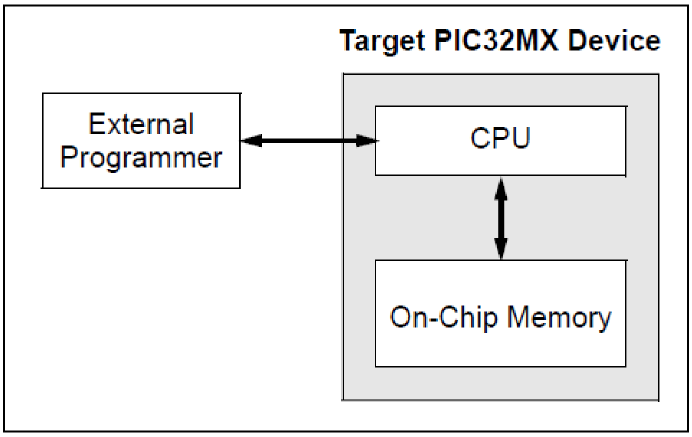


Figura 2.5. Programación ICSP en PIC32MX.

Teniendo en cuenta que disponíamos de dos herramientas de programación (ICD3 y PICKit3) se decidió elaborar la PCB con posibilidad de programar el micro con cualquiera de los dos programadores.

Las conexiones de ambos programadores al microcontrolador son las siguientes:

- 1. MCLR (Master Clear Pin).
- 2. $V_{DD} = 3.3V$.
- 3. V_{SS} (Ground).
- 4. PGED2. El PIC32MX795F512L dispone de dos pares de pines de programación (PGED1/PGEC1 y PGED2/PGEC2). Se seleccionó el segundo par de pines por comodidad a la hora de rutear la PCB. No hay diferencia entre ambos pares de pines y sólo se requerirá especificar en el código del programa cuáles de ellos se van a utilizar.
- 5.PGEC2.
- 6. No se utiliza.

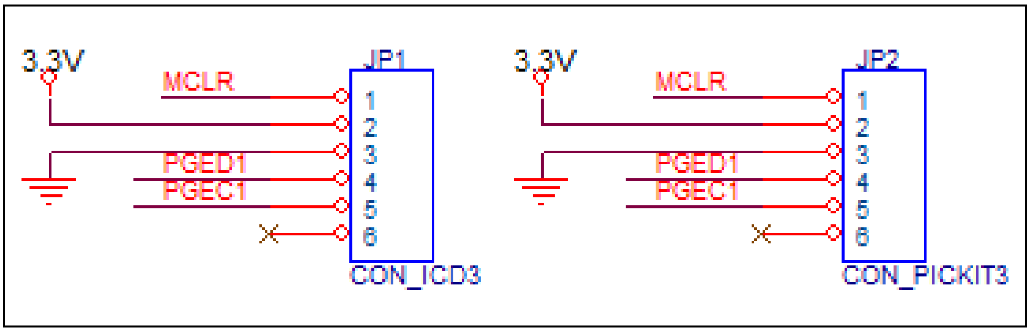


Figura 2.6. Circuito de programación ICSP.

Como podemos observar en el esquemático ambos programadores se conectan a los mismos pines del micro. Sin embargo, cada uno tiene un conector diferente y unas características específicas que se detallarán a continuación.

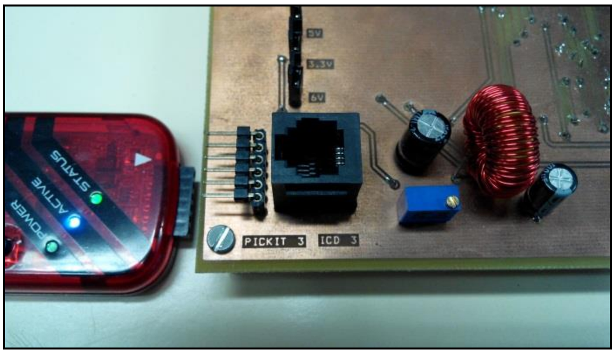


Figura 2.7. Imagen real de los conectores de programación.

2.3.3.1. PICKit 3

El PICKit 3 es la herramienta programación y depuración de bajo coste de Microchip..

Otra de las funcionalidades interesantes es la funcionalidad "Programmer-To-Go" que permite programar microcontroladores de la familia PIC16, PIC18, PIC24, dsPIC33F y PIC32 sin necesidad de conectarse a un ordenador. El PICKit 3 permite guardar un código de hasta 512KB en su Flash para programar directamente el PIC.

En este proyecto dado que solamente necesitamos programar un microcontrolador se ha utilizado esta herramienta para todo el proceso de programación y depuración.

El PICKit 3 tiene una sencilla conexión USB Full Speed que permite tanto programar y depurar como actualizar el firmware interno del PICKit.

Su conexión al microcontrolador se realiza mediante 6 conectores hembra situados a un lado del dispositivo. Por tanto se ha adquirido un conector en codo de 6 vías macho para la PCB.

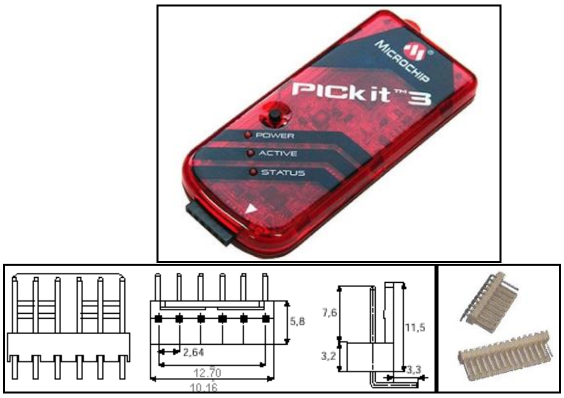


Figura 2.8. PICKit 3 y conectores utilizados en la PCB.

2.3.3.2. MPLAB ICD3

El MPLAB ICD3 es un programador y depurador en tiempo real de alta velocidad. Tiene como principales ventajas respecto al PICKit 3 la mayor velocidad en la programación de los microcontroladores así como la posibilidad de incluir un mayor número de puntos de ruptura en el código del programa.

Por tanto, en el caso de necesitar una programación en serie de programas con grandes cantidades de líneas de código sería conveniente utilizar el ICD3.

En nuestro caso se ha dotado al sistema de la posibilidad de conectarse a cualquiera de las dos herramientas de programación.

El conexionado del ICD3 a la tarjeta se realiza mediante un conector RJ-11 por lo que se ha adquirido un conector vertical de agujeros pasantes para facilitar su soldadura en la PCB de doble cara.

Figura 2.9. ICD 3 y conector utilizados en la PCB.

2.3.4. Oscilador.

En la arquitectura de los PIC32 existen dos buses diferentes para la comunicación de datos entre los diferentes componentes. El bus principal es utilizado por el núcleo del microcontrolador, por los periféricos con Acceso Directo a Memoria (DMA) y las interrupciones, entre otros. El resto de componentes trabajan a una frecuencia menor y se comunican a través del bus de periféricos.

Por la existencia de estos buses existen 3 relojes diferentes en la arquitectura del PIC32:

- El reloj del bus principal (SYSCLK)
- El reloj del bus de periféricos (PBCLK)
- El reloj destinado únicamente al USB (USBCLK)

Cada señal puede ser producida a su vez por diferentes fuentes de oscilación, en concreto 4:

- Oscilador externo primario (P_{OSC}) conectado entre los pines OSC1 y OSC2 del micro.
- Oscilador externo secundario (S_{OSC}) conectado entre los pines SOSCO y SOSCI.
- Oscilador RC interno (FRC) de 8MHz.
- Oscilador RC interno de baja potencia (LPRC) de 31.25kHZ.

En nuestro caso se ha decidido la colocación de un **oscilador externo primario (P_{OSC})** de 8MHz con el que conseguiremos mayor precisión que con el oscilador RC interno. El esquema electrónico para la conexión del oscilador externo sólo requiere de dos condensadores. El circuito se muestra a continuación.

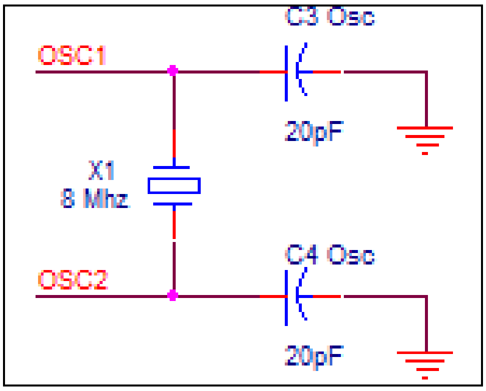


Figura 2.10. Circuito oscilador.

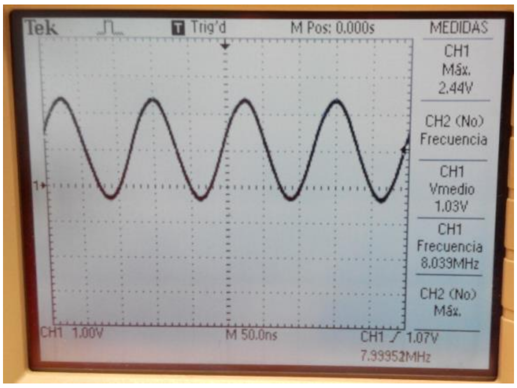


Figura 2.11. Señal entregada por el cristal externo de 8MHz.

2.3.5. Reset.

Otra de las funciones del Master Clear Pin (MCLR) además de la programación y la depuración del software es la de proporcionar un reset al microcontrolador. El reset por MCLR se consigue llevando momentáneamente el pin a un estado lógico bajo a través de la pulsación.

En nuestro caso se decidió no incluir un *reset por hardware*. Esto fue debido a que en todo momento se trabajó con el sistema conectado al ordenador para su depuración, permitiéndose así un *reset por software*.

En un sistema final sería conveniente incluir un reset por hardware a través de un pulsador. A continuación se muestra el circuito que sería necesario y el valor de los componentes del mismo para futuros proyectos.

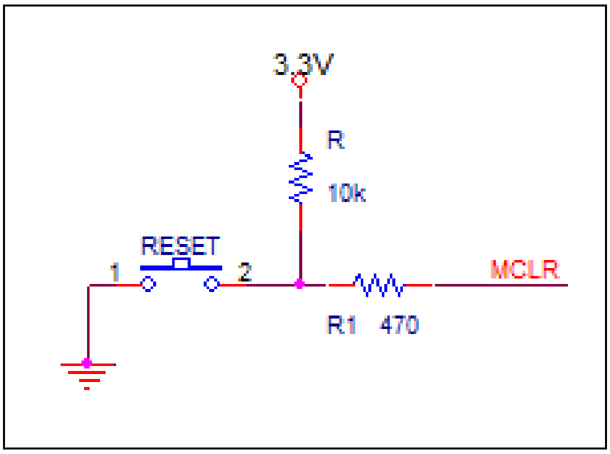


Figura 2.12. Conexión del botón de reset.

- $R \leq 10k\Omega$.
- $R1 \leq 470\Omega$.
- El pulsador se encargará de poner a nivel bajo el pin MCLR, con lo que se producirá el reinicio del dispositivo.

2.4. BLOQUE DE ALIMENTACIÓN

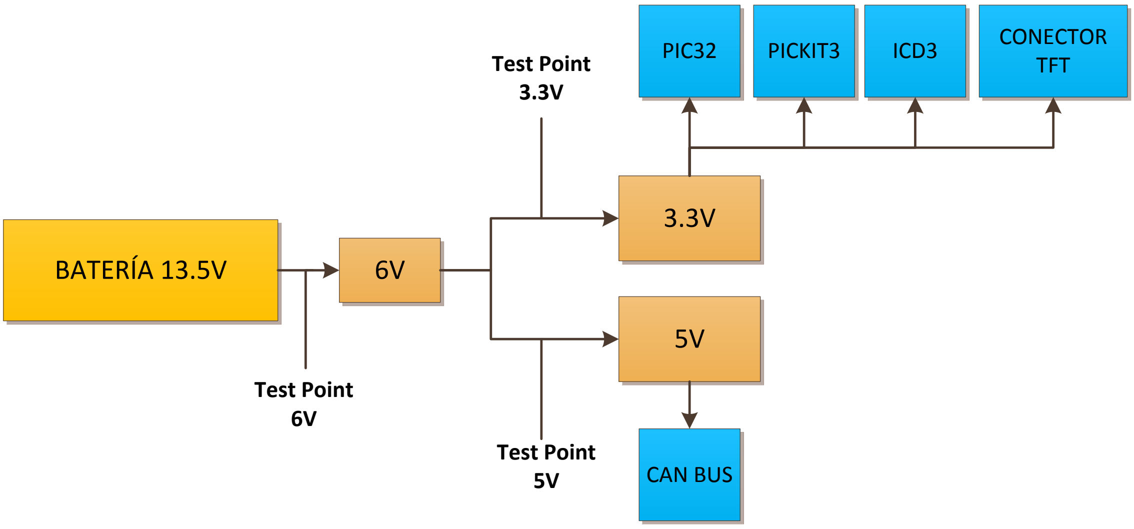


Figura 2.13. Diagrama de bloques. Bloque de alimentación.

Mediante el circuito de alimentación conseguimos proporcionar las tensiones estables necesarias para todos los componentes del sistema. La fuente de alimentación principal desde la que obtendremos todas las tensiones es la batería del coche de Fórmula SAE, que nos proporciona una salida no regulada de entre 12 y 13.5V.

Las salidas reguladas necesarias para el sistema son 5V (MCP2551 del CAN Bus) 3.3V (microcontrolador, programadores PICKit3, ICD3 y conexión a la tarjeta formada por la pantalla, el controlador gráfico y el panel táctil).

Además se requerirá de una tensión de 1.2V para el controlador gráfico SSD1963. En este caso la tarjeta adquirida que contiene la pantalla ya cuenta con un circuito regulador que convierte los 3.3V de entrada en 1.2V.

La regulación consta de dos fases: regulación conmutada y regulación lineal.

2.4.1. Regulación conmutada.

Previamente a la obtención de las salidas reguladas de 5V y 3.3V utilizamos un circuito **regulador conmutado** para bajar la tensión de entrada a un valor aceptable para los reguladores lineales utilizados en la siguiente fase.

Un regulador conmutado está construido con elementos reactivos como capacitores e inductores. Su principal virtud con respecto a los reguladores lineales es que no genera *pérdida de energía por disipación de calor* como sí ocurre con un regulador lineal. Por tanto transformaremos los 12-13.5V a 6V con un circuito regulador conmutado.

El regulador conmutado elegido es el **TL2575 ajustable** de Texas Instrument. Se trata de un regulador conmutado con una corriente máxima de salida de 1A. Con el circuito recomendado de su datasheet nos permite obtener a través de una **tensión de entrada no regulada de entre 7V y 40V** (en nuestro caso la batería del coche) una salida ajustable al valor deseado, en nuestro caso 6V. El circuito de salida ajustable recomendado es el siguiente

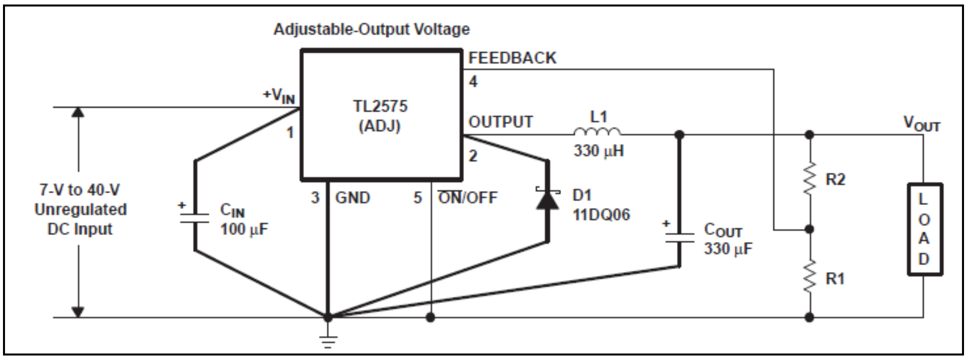


Figura 2.14. Circuito de ajuste con TL2575-ADJ.

Procedimiento de cálculo de compontes:

En el propio datasheet se nos indica el procedimiento de cálculo de componentes para la salida deseada, en este caso 6V.

Conocidos los siguientes datos:

$V_{OUT(Nom)} = 6V$

$V_{IN(Máx)} = 13.5V$ (La batería de alimentación nos da una salida no regulada entre 12V y 13.5V)

$I_{LOAD(Máx)} = 1A$

Selección de R1 y R2 conocido el voltaje de salida programable:

$V_{OUT} = V_{REF}\left(1 + \frac{R2}{R1}\right) \quad siendo \;\; V_{REF} = 1.23V$

Seleccionamos un valor de R1 entre 1kΩ y 5kΩ. Elegiremos

$\underline{\boldsymbol{R1 = 1k\Omega\ (1\%)}}$

Calculamos R2 según la ecuación siguiente

$$R2 = R1\left(\frac{V_{OUT}}{V_{REF}} - 1\right) = 3.878\text{k}\Omega$$

$$\underline{\boldsymbol{R2 = 3.83k\Omega\ \ (1\%)}}$$

Selección de la bobina (L1):

A. Cálculo del conjunto ($\mathrm{E \cdot T}$) que atraviesa L1.

$$\mathrm{E \cdot T} = (V_{IN} - V_{OUT}) \text{ x } t_{on}$$

$$\mathrm{E \cdot T} = (V_{IN} - V_{OUT}) \text{ x } (V_{OUT} / V_{IN}) \text{ x } \{1000/f_{osc}\,(en\ kHz)\}\ [V \cdot \mathrm{\mu} s]$$

$$\mathrm{E \cdot T} = (13.5 - 6) \text{ x } (6/13.5) \text{ x } \{1000/52\}\ [V \cdot \mathrm{\mu} s]$$

$$\mathrm{E \cdot T} = 64.102 [V \cdot \mathrm{\mu} s]$$

B. Una vez obtenido el valor de $\mathrm{E \cdot T}$ nos vamos a la figura 19 del datasheet y calculamos el valor referencia de L1.

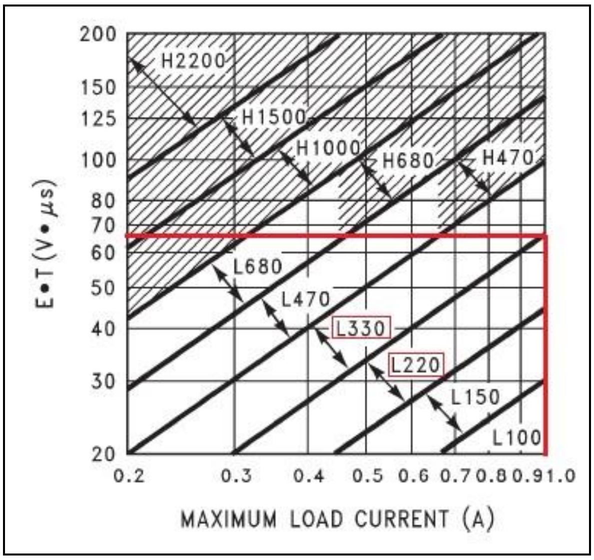


Figura 2.15. Cálculo de L1.

$$\underline{\boldsymbol{L1 = 330\ \mathrm{\mu} H}}$$

Selección del condensador de salida (C_{OUT}):

El bucle de control del TL2575 tiene una respuesta en frecuencia formada por dos polos y dos ceros. El par polo-cero dominante se establece con C_{OUT}y L1 a través de la siguiente expresión:

$$C_{OUT} \geq 7758\,\frac{V_{IN(Máx)}}{V_{OUT}\cdot L1(\mu H)}$$

$$C_{OUT} \geq 7758\,\frac{13.5}{6\cdot 330(\mu H)}$$

$$C_{OUT} \geq 52.89.\mu F$$

Para obtener un rizado aceptable de salida elegiremos el siguiente condensador

$$\underline{C_{OUT} = 220\ \mu F\ (electrolítico)}$$

Selección del diodo D1:

Para la elección de este diodo debemos ir a la tabla 1 del datasheet y seleccionar un diodo de 3A y una V_R de al menos 1.25 x $V_{IN(Máx)}$=16.875 por lo que escogeremos un diodo de similares características al 1N5820.

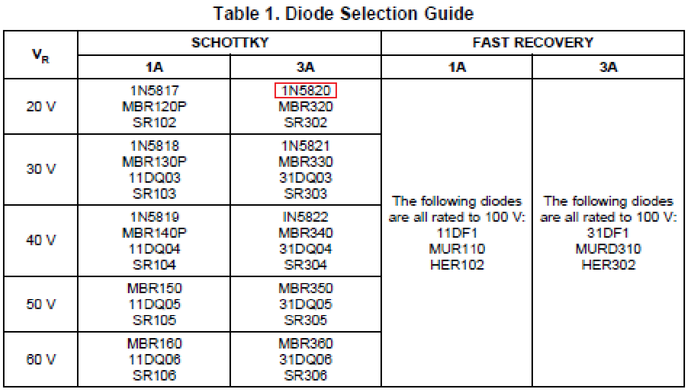


Table 1. Diode Selection Guide

V_R	SCHOTTKY		FAST RECOVERY	
	1A	3A	1A	3A
20 V	1N5817 MBR120P SR102	1N5820 MBR320 SR302	The following diodes are all rated to 100 V: 11DF1 MUR110 HER102	The following diodes are all rated to 100 V: 31DF1 MURD310 HER302
30 V	1N5818 MBR130P 11DQ03 SR103	1N5821 MBR330 31DQ03 SR303		
40 V	1N5819 MBR140P 11DQ04 SR104	IN5822 MBR340 31DQ04 SR304		
50 V	MBR150 11DQ05 SR105	MBR350 31DQ05 SR305		
60 V	MBR160 11DQ06 SR106	MBR360 31DQ06 SR306		

Figura 2.16. Selección del diodo D1.

$\underline{\boldsymbol{D1} = \textbf{Schottky SS32 de Multicomp}}$

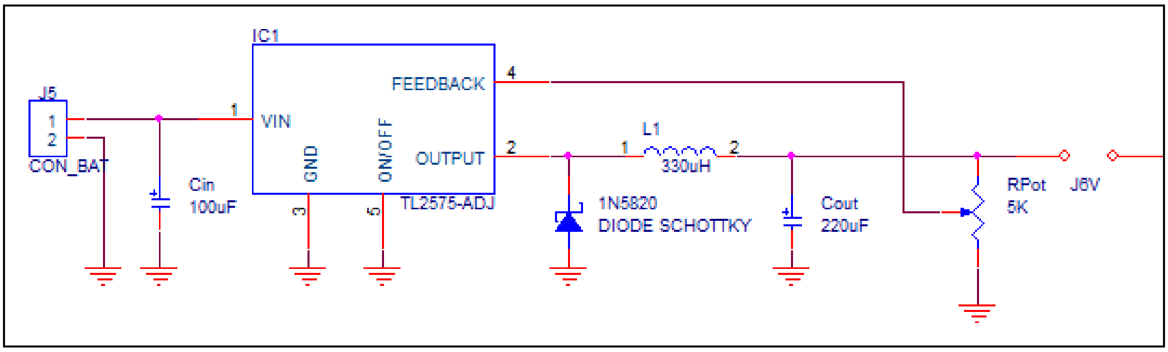


Figura 2.17. Circuito regulador conmutado.

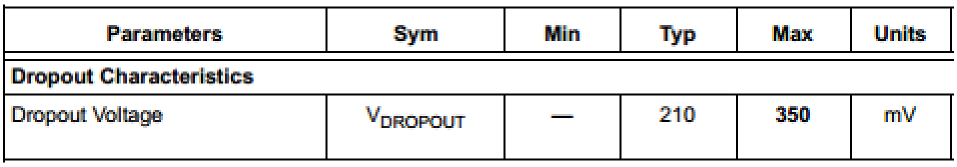
* Como se puede observar en el esquemático final se decidió añadir un potenciómetro de 5kΩ en lugar de las dos resistencias R1 y R2 para posibilitar un ajuste fino de la salida. *

2.4.2. Regulación lineal.

La siguiente fase después de haber obtenido una tensión de 6V es la obtención de las salidas reguladas de 5V y 3.3V. Para ello se seleccionan 2 reguladores lineales de bajo dropout (típicamente de 210mV y en el caso peor 350mV) de Microchip. En concreto **MCP1825S con salidas a 3.3V y 5V**.

Por tanto la tensión de entrada mínima de los reguladores será

$$V_{in} \geq V_{O(máx)} + V_{do(máx)}$$



Parameters	Sym	Min	Typ	Max	Units
Dropout Characteristics					
Dropout Voltage	$V_{DROPOUT}$	—	210	**350**	mV

Para el regulador de 3.3V

$$V_{in} \geq 3.3 + 350mV$$

$$V_{in} \geq 3.65V$$

Para el regulador de 5V

$$V_{in} \geq 5 + 350mV$$

$$V_{in} \geq 5.35V$$

En ambos casos con la entrada de 6V cumplimos la **condición de tensión de entrada**.

Además el fabricante nos indica que en ningún caso superemos los 6.5V de tensión de entrada para evitar la rotura del componente. Estos reguladores tienen una corriente de salida máxima de 500mA siendo más que suficiente para nuestro proyecto.

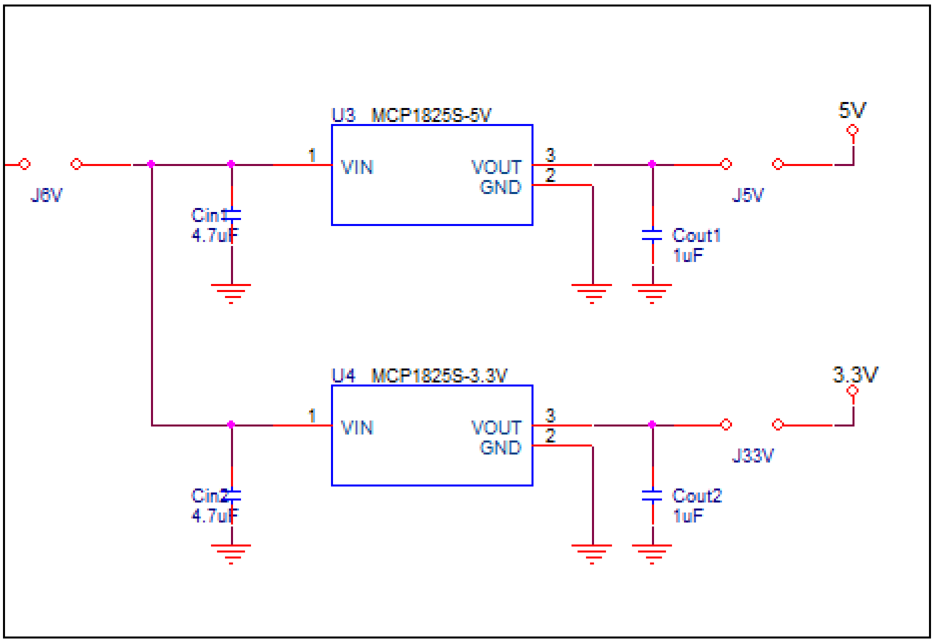


Figura 2.18. Circuito de regulación lineal.

2.5. CAN BUS.

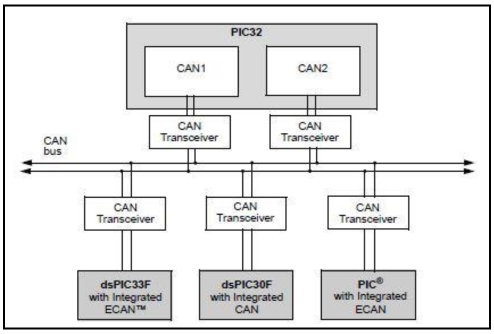


Figura 2.19. Sistema CAN Bus con diferentes PICs.

Algunos microcontroladores de Microchip cuentan con el controlador CAN integrado. Es el caso del micro utilizado en este proyecto, el PIC32MX795F512L. En nuestro caso contamos con dos módulos CAN. Por tanto para la implementación del nodo sólo necesitamos un transceiver para su conexión al bus.

El transceiver utilizado es el **MCP2551** de Microchip. Se trata de un transceiver de alta velocidad pudiendo operar hasta a velocidades de 1Mb/s, tolerante a fallos que servirá de interfaz entre el protocolo CAN y el bus físico.

Para gestionar el envío y la recepción de tramas CAN utilizaremos un simulador CAN proporcionado por el departamento. Para su conexión incluiremos en el circuito un conector DB-9 macho de los cuales conectaremos 3 pines (GND, CAN_H y CAN_L).

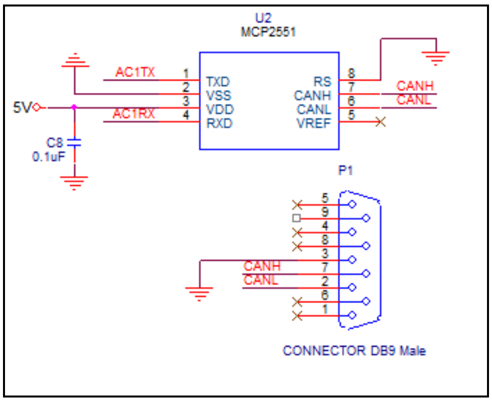


Figura 2.20. Circuito CAN Bus.

2.6. CONTROL GRÁFICO

2.6.1. Controlador gráfico SSD1963

El microcontrolador no puede conectarse directamente con la pantalla. Para ello necesita un controlador que se encargue de interpretar las señales recibidas y convertirlas en imágenes en pantalla. En nuestro caso utilizamos el controlador SSD1963 que viene incorporado junto a la pantalla y el panel táctil.

El SSD1963 es un controlador gráfico para pantallas LCD de alta resolución. Se encarga de generar las señales que necesita la pantalla de nuestro sistema. Este controlador será el único que tenga conexión directa con la pantalla. A su vez, nuestro microcontrolador se conectará al SSD1963 mediante el puerto paralelo PMP.

Estamos por tanto ante un controlador con gran capacidad de tratamiento de imágenes y por tanto una gran variedad de comandos y de opciones de configuración.

Las características del SSD1963 son las siguientes:

- RAM interna de 1215Kbytes con soporta de hasta 864 x 480 x 24bits.
- Soporte para TFT de 18/24 bits con interface RGB.
- Soporte serie de 8bits RGB.
- Rotación de imágenes en 0,90,180 y 270 grados.
- Efecto espejo de imagen.
- Posibilidad de almacenar varias ventanas de visualización.
- Brillo, contraste y saturación programable.
- Control de iluminación programable vía PWM.
- Conectividad de 8/9/16/18/24 bits a microcontroladores.
- 4 puertos de entrada/salida de propósito general.
- Generador de señal de reloj independiente.
- Modo de ahorro de energía.

- Requiere de doble alimentación regulada: 1.2V y 3.3V típicamente.

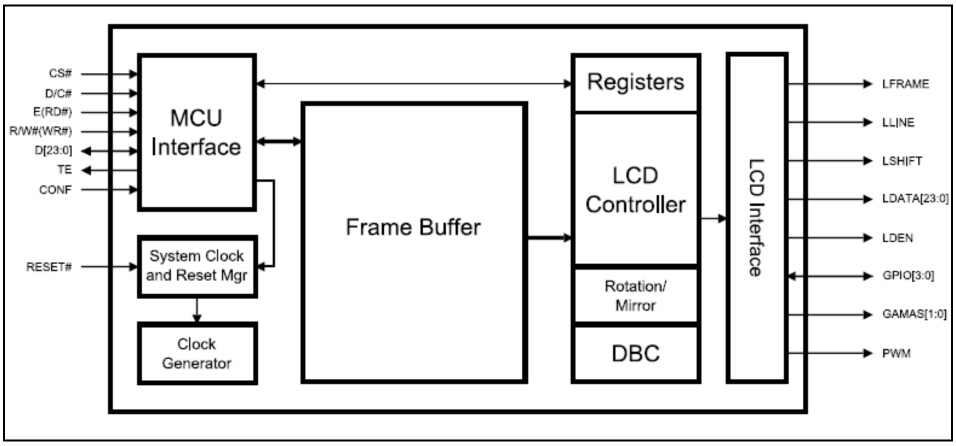


Figura 2.21. Diagrama de bloques del SSD1963.

2.6.2. Puerto Paralelo Maestro (PMP).

El Puerto Paralelo Maestro (PMP) es un módulo paralelo de entradas y salidas de 8 ó 16 bits especialmente diseñado para una amplia variedad de dispositivos como periféricos de comunicación, LCDs, dispositivos de memoria externa y microcontroladores. Debido a la amplia variedad de dispositivos que pueden utilizarse es altamente configurable.

Las características del módulo PMP de los PIC incluyen:

- Hasta 16 líneas de dirección programables.
- Hasta 2 líneas de selección de componente.
- Opciones de programación tanto de escritura y lectura individual como mediante habilitación.
- Auto incremento y auto decremento de dirección.
- Dirección programable y multiplexación de datos.
- Polaridad programable en las señales de control.

- Soporte del esclavo paralelo mejorado.
- Estados de esperas programables.
- Opción de congelación en programación in-circuit.

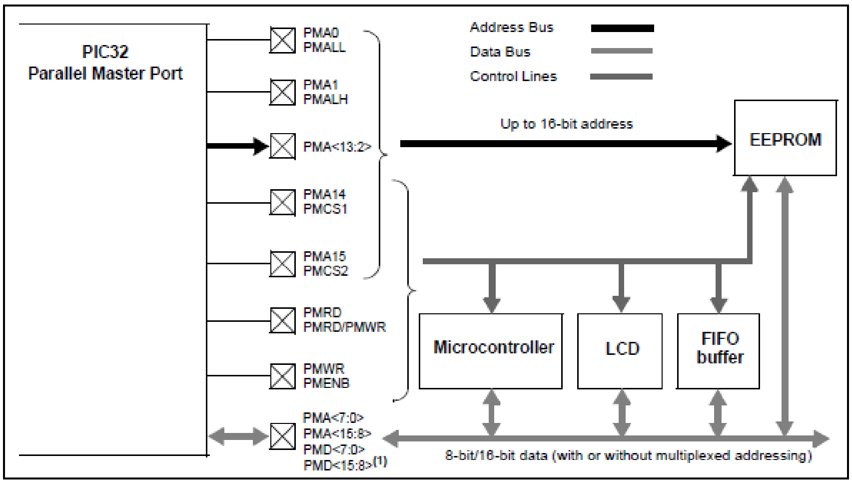


Figura 2.22. Diagrama de bloques del Puerto Paralelo Maestro (PMP) en un PIC32.

Las conexiones necesarias para la conexión y comunicación entre el PIC con la tarjeta que contiene la pantalla, y más concretamente con el controlador gráfico SSD1963, son las siguientes:

- PMD[0:15] : Bus de datos.
- PMRD: Señal de activación de lectura.
- PMWR: Señal de activación de escritura.
- RS: Indica si se va a enviar un dato o un comando al SSD1963.
- PMCS1: Chip Select.
- REST: Reset.

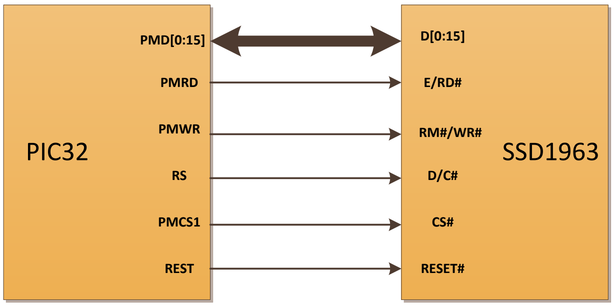


Figura 2.23. Conexión del PMP entre PIC32 y SSD1963.

2.6.3. CONEXIÓN DE SSD1963 Y PANTALLA.

La conexión del controlador SSD1963 con la pantalla viene incluida en el conjunto adquirido. Aun así se comentarán las conexiones necesarias en caso de quererse conectar otra pantalla en futuros proyectos.

Las conexiones necesarias en nuestro caso son las siguientes:

- 24 líneas de datos RGB888 que corresponden a 8 líneas por cada color primario (rojo, verde, azul).
- DCKL, HSYNC, VSYNC: señales de reloj y de sincronización horizontal y vertical dependientes de la pantalla con la que trabajemos.
- LCD_LDEN: señal de dato válido.

En el sistema adquirido se conectan varios sistemas mediante un cable plano que contiene información del conjunto del panel táctil (incluyendo pantalla, panel táctil y retroiluminación). El resto de conexiones del cable plano (que no pertenecen al controlador gráfico) son:

- VLED+ y VLED- procedentes del sistema de retroiluminación.
- X-,X+,Y-,Y+ : señales del panel táctil que se procesarán en el circuito correspondiente al control táctil.

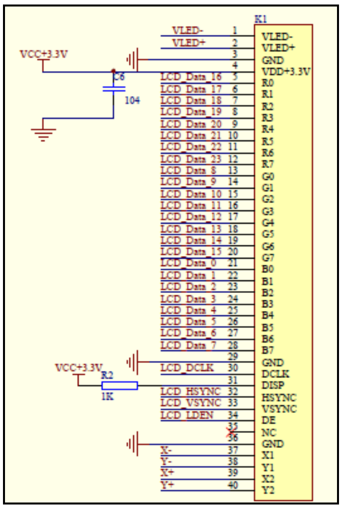


Figura 2.24. Conexión de SSD1963 con pantalla.

2.7. CONTROL TÁCTIL

El panel táctil incorporado a la pantalla es de tipo resistivo de 4 hilos (para más información sobre los paneles resistivos de 4 hilos consultar el capítulo de introducción dedicado a los paneles táctiles). Mediante el circuito de control táctil seremos capaces de traducir e interpretar la información analógica recibida por el panel a través de sus 4 líneas (X+,Y+, X-, Y-) para acabar obteniendo las coordenadas digitales X e Y.

Para todo este proceso requeriremos de un controlador para paneles táctiles resistivos. El controlador táctil utilizado es el XPT2046 que ya incluía el módulo adquirido. Nos comunicaremos a través del microcontrolador por vía SPI.

2.7.1. XPT2046.

El XPT2046 es un controlador táctil para paneles resistivos de 4 hilos cuyas principales características son las siguientes:

- Incorpora un convertidor analógico-digital de 12 bits por aproximaciones sucesivas (SAR). El funcionamiento de este tipo de convertidores se basa en un proceso iterativo, mediante el cual, se van probando diferentes códigos digitales hasta encontrar el correspondiente a la señal analógica de entrada.

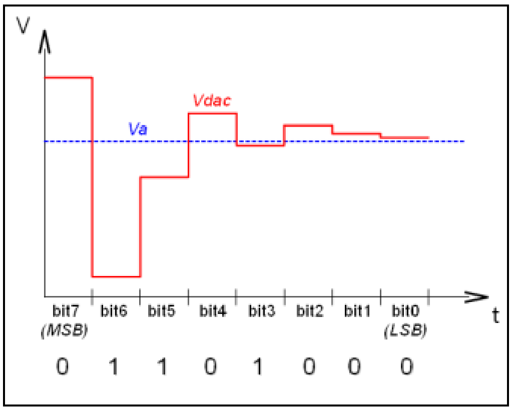


Figura 2.25. ADC por aproximaciones sucesivas (SAR).

- Su frecuencia de muestreo máxima es de 125kHz.
- Cuenta con un medidor de presión y otro de temperatura.
- Bajo consumo (260µA).

La conexión del XPT2046 con el micro y el panel es la siguiente:

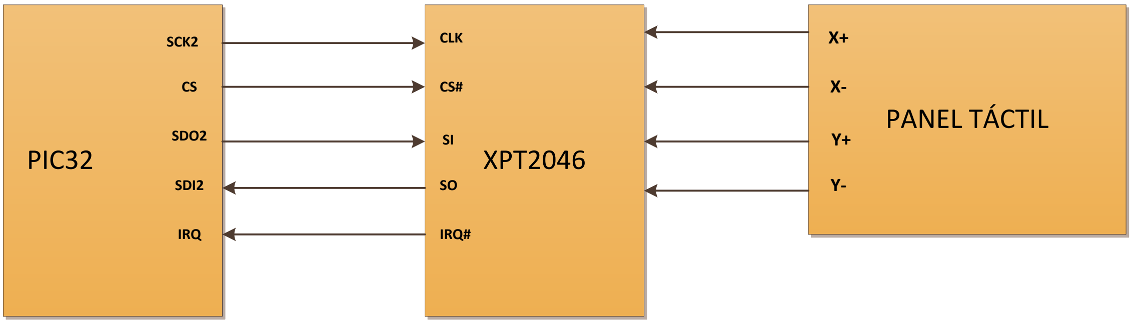


Figura 2.26. Diagrama de bloques y conexiones a panel táctil.

2.7.2. Comunicación SPI.

La comunicación SPI (Serial Peripheral Interface) es un protocolo de comunicaciones síncrono utilizado para la transferencia de información entre varios circuitos integrados. Pese a que existen otros métodos con menos conexiones el SPI sólo requiere de tres conexiones mínimas. Ésto le permitie mayor integración de componentes electrónicos que en una comunicación paralelo. A pesar de tratarse de una transmisión serie su velocidad es mayor que otras conexiones serie como el I2C por ejemplo.

La interfaz SPI se basa en la existencia de un maestro que será el encargado de suministrar el reloj de sincronización a la conexión y uno o varios esclavos. En el caso de utilizarse varios esclavos se necesitará una señal adicional (SS) para la selección del esclavo utilizado en ese momento.

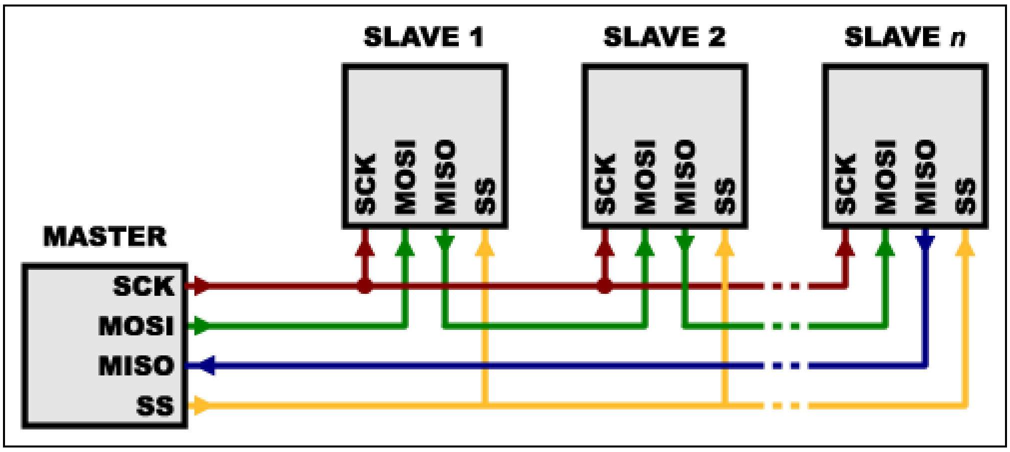


Figura 2.27. Comunicación SPI Maestro-Esclavo.

En nuestro caso el PIC32 hará la labor de maestro mientras que el XPT2046 será el esclavo.

El PIC32MX795 utilizado en este proyecto cuenta con cuatro módulos SPI. Para facilitar el ruteado se ha decidido la utilización del SPI2 además de dos puertos de entrada/salida estándar. Las conexiones y su correspondencia con el integrado XPT2046 se detallan a continuación (entre paréntesis se indica el nombre de la conexión en el esquemático).

- SCK2 (T_CLK): Reloj serie de sincronía entre maestro y esclavo. Pin de entrada-salida.
- SDI2 (T_DO): Entrada de datos de SPI2. Pin de entrada al PIC y de salida del XPT2046.
- SDO2 (T_DIN): Salida de datos de SPI2. Pin de salida del PIC y de entrada del XPT2046.
- RB4 (T_CS) : Chip Select del XPT2046.Pin de entrada-salida estándar.
- RC13 (T_IRQ): Pin de salida que indica que se ha realizado una pulsación.

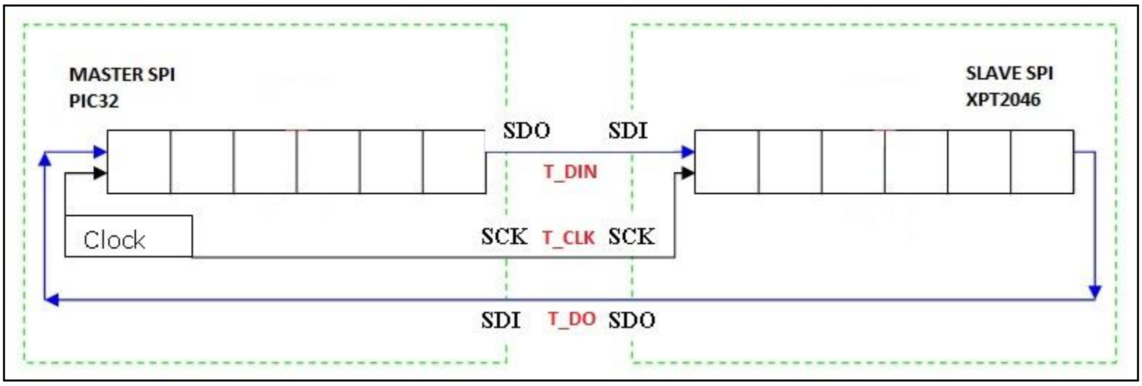


Figura 2.28. Comunicación SPI entre el PIC32 y el XPT2046.

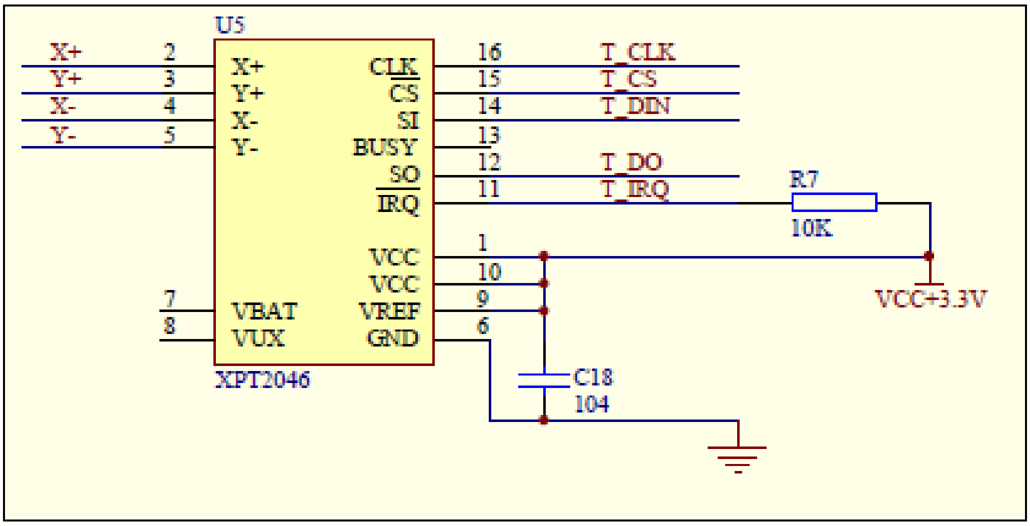


Figura 2.29. Circuito de conexión SPI.

2.8. CONTROL DE BRILLO

Una de las particularidades que tiene el sistema adquirido es la retroiluminación LED de la pantalla que sustituye los tubos fluorescentes utilizados años atrás. Mediante el uso de la tecnología LED conseguimos varios beneficios, entre los que destacan la **reducción del consumo** y la capacidad de crear **pantallas más delgadas**.

Dentro de la tecnología de retroiluminación LED existen varias alternativas. La más común y la que utiliza la pantalla utilizada en el proyecto es conocida como **iluminación perimetral**. La iluminación perimetral consiste en la colocación de leds en los laterales de la pantalla que consiguen un brillo uniforme debido a un diseño especial de difusores.

La otra alternativa, mucho más costosa es la creación de una **matriz de leds** a lo largo de toda la superficie de la pantalla, permitiendo incluso la iluminación independiente de todos ellos consiguiendo así mayor contraste entre negros y blancos.

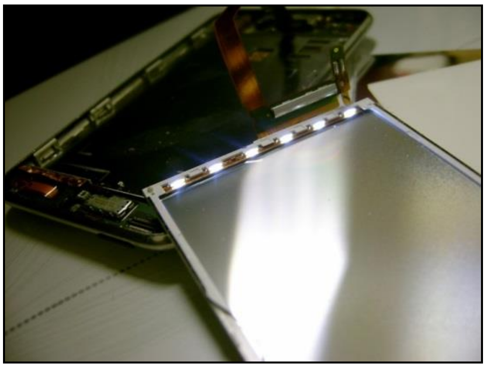


Figura 2.30. Iluminación perimetral en una pantalla.

Debido a que se trata de un modelo comercial el fabricante nos permite la gestión de la retroiluminación de dos maneras distintas gracias al jumper 3:

- **Iluminación máxima** sin posibilidad de configuración. Se consigue dejando el jumper 3 en cortocircuito. Esto hará que el voltaje de entrada al integrado sea siempre de 3.3V consiguiendo así la iluminación máxima posible en el dispositivo.

- **Iluminación variable mediante modulación por ancho de pulsos (PWM).** Dejando J3 en circuito abierto tendremos dos entradas distintas al integrado. Una de ellas (VIN) será siempre 3.3V, y la otra (EN) provendrá de la señal proveniente del PWM del controlador gráfico con el ciclo de trabajo deseado en cada momento. Las señales provenientes del PWM tendrán un ciclo de trabajo variable, y por tanto, su valor medio también variará. Mediante esta señal aplicada a la patilla EN del integrado TPS61040 y la señal de entrada de 3.3V conseguiremos el ajuste del brillo de la pantalla.

Se ha elegido la segunda opción para añadir en las opciones accesibles al piloto una opción de control de brillo.

El siguiente esquemático corresponde al circuito electrónico que gestiona la retroiluminación de nuestro sistema. El circuito utiliza el integrado **TPS61040**. El TPS61040 es un **convertidor DC/DC de baja potencia** utilizado para la alimentación en aplicaciones de retroiluminación LED de pequeños dispositivos donde es necesario una excitación uniforme de los leds. Sus características principales son las siguientes:

- Rango de tensión de entrada de 1.8V a 6V.
- Capaz de generar voltajes elevados de hasta 28V.
- Corriente máxima limitada a 400mA.

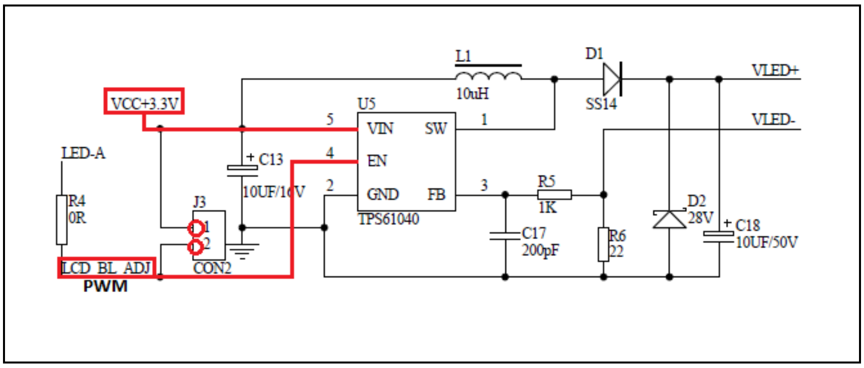


Figura 2.31. Conexión del circuito de retroiluminación LED.

3. DISEÑO Y CREACIÓN DE PCB.

El *diseño de una PCB* requiere de dos fases claramente diferenciadas utilizando herramientas de diseño asistido por computadora (CAD). En la primera de ellas implementaremos nuestro esquema electrónico previamente diseñado en una herramienta CAD. Posteriormente importaremos este esquema en una herramienta de diseño de PCBs.

Es importante recalcar que tanto la herramienta de diseño de esquemáticos como la herramienta de diseño de PCBs deben ser compatibles. Por tanto, se ha decidido utilizar el paquete de herramientas Orcad en su versión 10.5, que contiene el **Orcad Capture** para crear esquemas electrónicos y el **Orcad Layout** para el diseño de PCBs.

La *fabricación de la PCB*, una vez obtenidos los fotolitos propios del sistema, se procede a la fabricación utilizando los medios disponibles por el departamento.

3.1. Captura de esquemas. Orcad Capture CIS. Ultra Librarian.

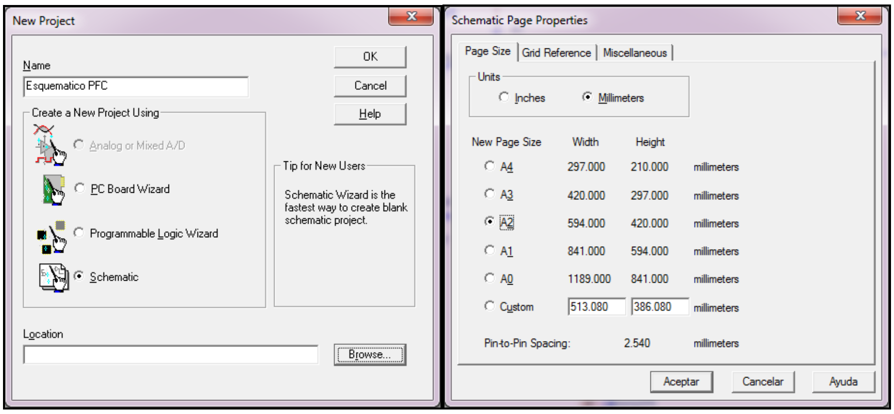
El esquema electrónico del circuito es una representación simbólica de los componentes que forman un circuito. Cada símbolo que compone en esquema es una representación abstracta del componente. En esta etapa del diseño sólo es importante la conexión entre los componentes, por ello la fidelidad del símbolo del componente con el componente real es meramente estética. Además de herramienta de captura de esquemas Orcad Captura se ha utilizado Ultra Librarian

3.1.1. Fases de la creación del esquema electrónico en Orcad Capture.

1.- Creación del proyecto y configuración previa.

Seleccionaremos File->New->Project... y nos aparecerá la siguiente ventana en la que tendremos varias opciones a elegir. Seleccionaremos ahora Schematic, daremos nombre a nuestro proyecto y elegiremos la ruta de guardado.

Una vez hecho esto, nos aparecerá la página principal del esquemático donde colocaremos los componentes, pero antes de ello deberemos configurar un tamaño mayor de página desde Options->Schematic Page Properties... . En nuestro caso hemos seleccionado un tamaño A2 para poder situar todos los subcircuitos en la misma página de proyecto.



New Project

Name: Esquematico PFC

Create a New Project Using

○ Analog or Mixed A/D
○ PC Board Wizard
○ Programmable Logic Wizard
● Schematic

Tip for New Users

Schematic Wizard is the fastest way to create blank schematic project.

Location:

Schematic Page Properties

Units: ○ Inches ● Millimeters

New Page Size	Width	Height	
○ A4	297.000	210.000	millimeters
○ A3	420.000	297.000	millimeters
● A2	594.000	420.000	millimeters
○ A1	841.000	594.000	millimeters
○ A0	1189.000	841.000	millimeters
○ Custom	513.080	386.080	millimeters

Pin-to-Pin Spacing: 2.540 millimeters

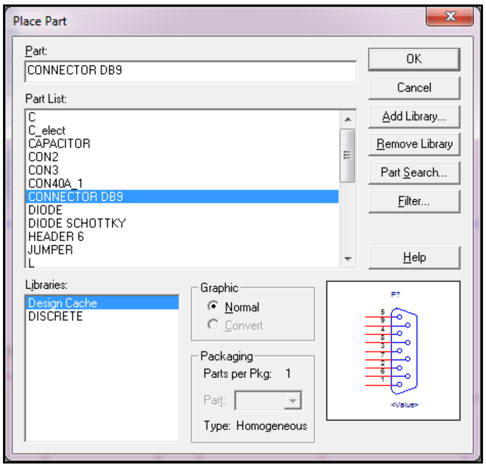
Figura 3.1. Primeros pasos en proyecto de Orcad Capture.

2.- Colocación de compontes.

Con la configuración previa realizada, ya podemos comenzar a situar los componentes. Los componentes se añaden en Place -> Part o en su icono correspondiente en la barra de herramientas.

Nos aparecerá la ventana mostrada a continuación. Orcad Capture nos proporciona un conjunto extenso de librerías en las que encontraremos los componentes más comunes utilizados en los circuitos electrónicos, así como componentes más específicos.

También colocaremos componentes comunes a todos los circuitos que poseen de icono propio en la barra de tareas como son VCC PWR , GND GND o el de PIN NO CONECTADO , entre otros.



Place Part

Part: CONNECTOR DB9

Part List:
C
C_elect
CAPACITOR
CON2
CON3
CON40A_1
CONNECTOR DB9
DIODE
DIODE SCHOTTKY
HEADER 6
JUMPER
L

Libraries:
Design Cache
DISCRETE

Graphic
● Normal
○ Convert

Packaging
Parts per Pkg: 1
Part:
Type: Homogeneous

Figura 3.2. Colocación de componentes. Orcad Capture.

Sin embargo, para nuestro proyecto necesitaremos otros componentes que no aparecen en dichas librerías. Es el caso de algunos componentes de Microchip como el propio microcontrolador o los reguladores lineales. Para ello utilizaremos un programa llamado **Ultra Librarian** que nos proporcionará tanto el símbolo del componente para el esquemático como la huella, evitando así posteriormente tener que crear la huella manualmente de estos componentes.

A medida que introduzcamos cada componente es recomendable asignarles un valor y un nombre para su posterior identificación en la siguiente etapa del proceso de colocación de componentes en la placa de circuito impreso.

3.- Conexiones eléctricas.

Una vez situados todos los componentes realizaremos las uniones eléctricas de los mismos con Place -> Wire o su botón correspondiente. Otra opción es la utilización de etiquetas para facilitar la lectura y comprensión del esquemático, otorgando mayor limpieza al esquema.

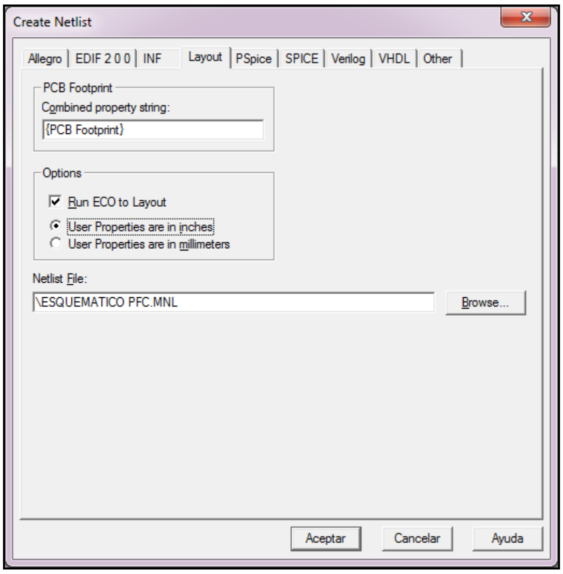
4.- Asignación de huellas de componentes (footprints).

A continuación, y previamente a la edición y creación física de la placa de circuito impreso, asignaremos a cada componente su footprint o marca física sobre el circuito. Para ello podemos asignar el footprint previamente desde Orcad Capture seleccionando todo el circuito con cntl+A y en Edit properties.../Parts asignando uno a uno el footprint del componente. De otro modo, una vez iniciado el layout se nos indicarán aquellos componentes que no disponen de footprint y serán asignados en ese momento.

Como ocurre en el caso de los símbolos de componentes, no todas las huellas se encuentran en las librerías básicas de footprints. Algunas de ellas se han tenido que crear manualmente y otras obtenerlas del programa Ultra Librarian.

5.- Creación de netlist.

Una vez completemos el esquemático es el momento de darle un formato comprensible por Orcad Layout. Para ello desde la ventana de proyecto con el archivo .dsn activado pulsaremos en el botón de la barra de herramientas Create Netlist . En la siguiente ventana, desde la pestaña Layout seleccionaremos Run ECO to Layout y User Properties are in inches. Pulsaremos en aceptar y se nos habrá creado el netlist que utilizaremos posteriormente en el diseño de la PCB.



Combined property string: {PCB Footprint}

☑ Run ECO to Layout

● User Properties are in inches

○ User Properties are in millimeters

Netlist File: \ESQUEMATICO PFC.MNL

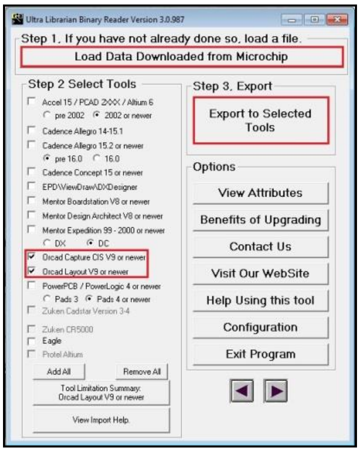
Figura 3.3. Creación de NetList. Orcad Capture.

3.1.2. Obtención de símbolos y huellas. Ultra Librarian.

Ultra Librarian es una herramienta compartida por Microchip que nos facilita el desarrollo de un proyecto proporcionándonos acceso a huellas y símbolos CAD/CAE de productos Microchip evitándonos tener que crearlos manualmente.

Para ello seguiremos los siguientes pasos:

1. Descargaremos e instalaremos el programa Ultra Librarian. Podemos encontrarlo en la web de Microchip dentro de la ruta Design Support/ Design & Simulation Tools/ CAD/CAE Symbols.
2. Desde la misma página descargaremos los archivo 'bxl' de los componente que deseemos, en nuestro caso del microcontrolador *PIC32MX795F512L*, del controlador Can *MCP2551* y de los reguladores lineales *MCP1825S*.
3. Una vez dentro del programa, cargaremos el archivo del componente, seleccionaremos las herramientas en las que queremos que se nos generen las huellas/símbolos (en nuestro caso serán Orcad Capture y Orcad Layout) y pulsaremos en el botón de exportar.



Ultra Librarian Binary Reader Version 3.0.987

Step 1. If you have not already done so, load a file.

Step 2 Select Tools

☐ Accel 15 / PCAD 2XXX / Altium 6
○ pre 2002 ● 2002 or newer
☐ Cadence Allegro 14-15.1
☐ Cadence Allegro 15.2 or newer
● pre 16.0 ○ 16.0
☐ Cadence Concept 15 or newer
☐ EPD\ViewDraw\DXDesigner
☐ Mentor Boardstation V8 or newer
☐ Mentor Design Architect V8 or newer
☐ Mentor Expedition 99 - 2000 or newer
○ DX ● DC
☑ Orcad Capture CIS V9 or newer
☑ Orcad Layout V9 or newer
☐ PowerPCB / PowerLogic 4 or newer
○ Pads 3 ● Pads 4 or newer
☐ Zuken Cadstar Version 3-4
☐ Zuken CR5000
☐ Eagle
☐ Protel Altium

Tool Limitation Summary:
Orcad Layout V9 or newer

Step 3. Export

Options

Figura 3.4. Página principal. Ultra Librarian.

4. Se nos creará un archivo .EDF en el caso de Orcad Capture y otro .MIN en el caso de Orcad Layout para importar el símbolo y la huella de los componentes desde los mismos.

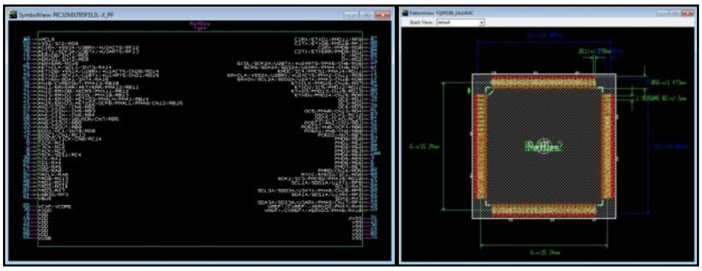


Figura 3.5. Símbolo y huella. PIC32MX795F512L.

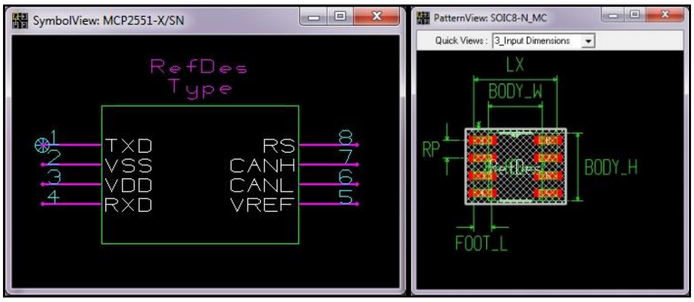
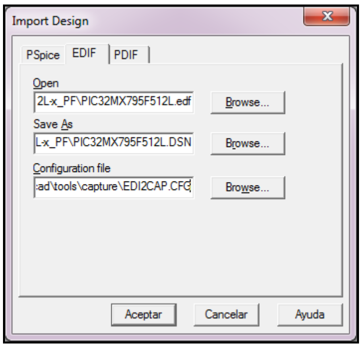


Figura 3.6. Símbolo y huella generados de un MCP2551.

5. Para la importación del símbolo en el caso de **Orcad Capture** deberemos seleccionar File-> Import Design... y nos movemos a la pestaña EDIF.

   Seleccionaremos el archivo .EDF creado previamente en la opción open. En la última opción deberemos buscar el archivo EDI2CAP.cfg situado en la carpeta de instalación de Orcad Capture.

Una vez pulsemos aceptar se nos habrá creado una librería de capture (.olb) que ya podremos utilizar en Place-> Part como cualquier otra librería de Orcad.



Open: 2L-x_PF\PIC32MX795F512L.edf

Save As: L-x_PF\PIC32MX795F512L.DSN

Configuration file: :ad\tools\capture\EDI2CAP.CFG

Figura 3.7. Importación de un componente. Ultra Librarian.

Para la importación del símbolo en el caso de **Orcad Layout** seleccionamos File-> Import-> MIN Interchange y elegiremos los archivos .min creados previamente por Ultra Librarian. Posteriormente entre las opciones disponibles elegiremos .llb que será el formato de librería admitido por Orcad Layout.

3.1.3. Creación manual de huellas de componentes.

Tanto para la creación manual de footprints como para buscar huellas genéricas, accederemos al Library Manager de Orcad Layout desde la pestaña Tools. Utilizaremos como ejemplo uno de los footprints que hemos tenido que crear manualmente correspondiente al cristal del sistema.

Para comenzar a crear la creación de la footprint necesitaremos conocer los datos precisos de las medidas del componente, por ello utilizaremos la sección del datasheet del componente relativa a las dimensiones del mismo.

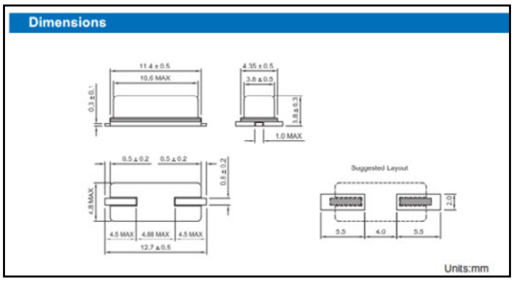


Figura 3.8. Dimensionado de un componente.

Seleccionaremos Create New Footprint , daremos nombre a la huella y seleccionaremos las medidas en las que vamos a trabajar dependiendo de los datos aportados por el fabricante. En este caso las medidas vienen dadas en milímetros.

Figura 3.9. Creación de una nueva huella.

Comenzaremos creando los pines del componente. Como observamos en la imagen anterior, se nos habrá creado un pin circular que deberemos editar. Vamos a crear la huella de un componente de montaje superficial (SMD) por lo que los pines son rectangulares.

Para editar el pin pulsaremos en el icono de View Spreadsheet ▦ y después en el desplegable Padstacks, que contiene toda la información del tamaño, forma de cada una de las capas del footprint.

Editaremos la capa Top cambiando la forma del pin por un rectángulo de las dimensiones que nos recomienda el fabricante (5.5 x 2 mm). Copiaremos el pin ya creado y crearemos otro con una distancia de 4mm entre ellos.

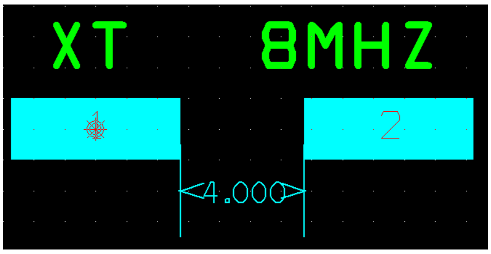


Figura 3.10. Creación de los pads.

Además de los pads del componente, hemos añadido desde Obstacle Tool información sobre el tamaño del componente, así como un borde exterior en el componente.

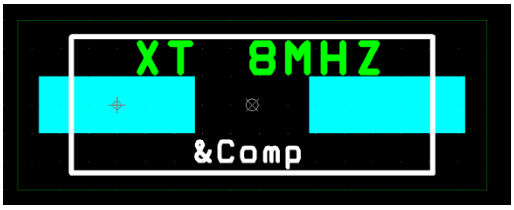
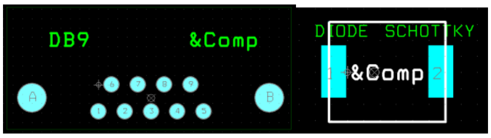
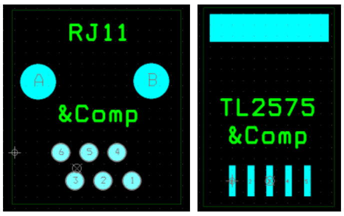
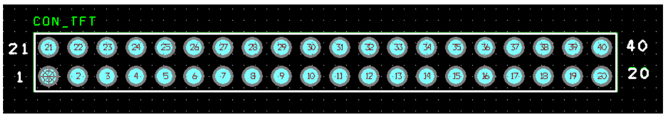


Figura 3.11. Huella del cristal de 8MHz.

A continuación se muestran otras huellas que se han tenido que crear para la elaboración de circuito impreso.



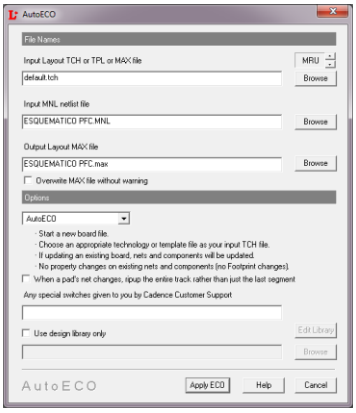




3.2. Diseño físico de PCB. Orcad Layout.

A continuación procederemos a diseñar la placas de circuito impreso con ayuda de la herramienta de Orcad Layout Plus 10.5.

Comenzaremos abriendo un nuevo proyecto con File->New y se nos desplegará la siguiente ventana. La rellenaremos con la ruta de default.tch en el primer lugar, del netlist creado previamente con Capture en el segundo y con el nombre que queramos dar al fichero .MAX de Layout en el último.



AutoECO

File Names

Input Layout TCH or TPL or MAX file: default.tch

Input MNL netlist file: ESQUEMATICO PFC.MNL

Output Layout MAX file: ESQUEMATICO PFC.max

☐ Overwrite MAX file without warning

Options

AutoECO

- Start a new board file.
- Choose an appropriate technology or template file as your input TCH file.
- If updating an existing board, nets and components will be updated.
- No property changes on existing nets and components (no Footprint changes).

☐ When a pad's net changes, ripup the entire track rather than just the last segment

Any special switches given to you by Cadence Customer Support

☐ Use design library only

Figura 3.12. Orcad Layout.

Si hemos identificado previamente todos los footprints correctamente nos aparecerán todos los componentes en la ventana de edición así como las conexiones entre ellos en líneas amarillas. En caso contrario nos indicará un mensaje emergente que añadamos el footprint del componente que nos falte.

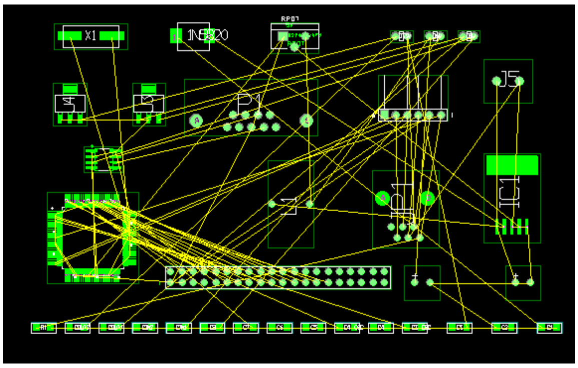


Figura 3.13. Colocación de componentes. Orcad Layout.

Antes de comenzar con el ruteo de la placa debemos establecer el borde de la placa. Para ello debemos crear un obstáculo. Pulsaremos en el botón Obstacle Tool de la barra de herramientas y en las propiedades de obstáculo seleccionaremos Board Outline, el ancho de línea y Global Layer. A continuación haremos un rectángulo que englobe a todos los componentes.

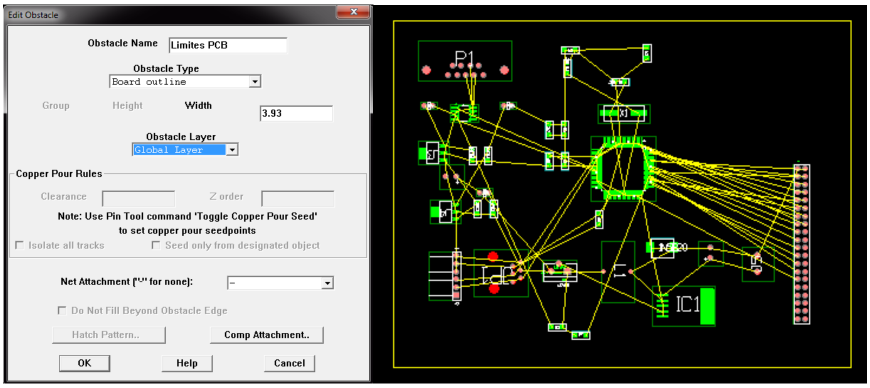
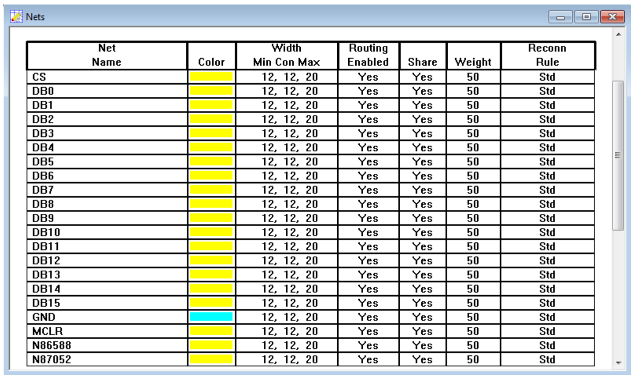

Obstacle Name: Limites PCB
Obstacle Type: Board outline
Width: 3.93
Obstacle Layer: Global Layer
Note: Use Pin Tool command 'Toggle Copper Pour Seed' to set copper pour seedpoints
Net Attachment ("-" for none): -

Figura 3.14. Borde de la PCB. Orcad Layout.

A continuación asignaremos el ancho mínimo y máximo de las pistas del diseño. Pulsaremos en Window/Nets. Debido a la cantidad de conexiones que tiene el microcontrolador y la distancia entre los 100 pines de los que dispone, asignaremos una anchura de pista mínima de 12mils. En el resto del circuito no tendremos tantos problemas de ruteo por lo que pondremos un tamaño máximo de 20 mils.



Net Name	Color	Width Min Con Max	Routing Enabled	Share	Weight	Reconn Rule
CS	🟨	12, 12, 20	Yes	Yes	50	Std
DB0	🟨	12, 12, 20	Yes	Yes	50	Std
DB1	🟨	12, 12, 20	Yes	Yes	50	Std
DB2	🟨	12, 12, 20	Yes	Yes	50	Std
DB3	🟨	12, 12, 20	Yes	Yes	50	Std
DB4	🟨	12, 12, 20	Yes	Yes	50	Std
DB5	🟨	12, 12, 20	Yes	Yes	50	Std
DB6	🟨	12, 12, 20	Yes	Yes	50	Std
DB7	🟨	12, 12, 20	Yes	Yes	50	Std
DB8	🟨	12, 12, 20	Yes	Yes	50	Std
DB9	🟨	12, 12, 20	Yes	Yes	50	Std
DB10	🟨	12, 12, 20	Yes	Yes	50	Std
DB11	🟨	12, 12, 20	Yes	Yes	50	Std
DB12	🟨	12, 12, 20	Yes	Yes	50	Std
DB13	🟨	12, 12, 20	Yes	Yes	50	Std
DB14	🟨	12, 12, 20	Yes	Yes	50	Std
DB15	🟨	12, 12, 20	Yes	Yes	50	Std
GND	🟦	12, 12, 20	Yes	Yes	50	Std
MCLR	🟨	12, 12, 20	Yes	Yes	50	Std
N86588	🟨	12, 12, 20	Yes	Yes	50	Std
N87052	🟨	12, 12, 20	Yes	Yes	50	Std

Figura 3.15. Edición de pistas. Orcad Layout.

Una vez seleccionado el ancho de pista es momento de indicar que capas queremos rutear desde Tool/Layer/ Select from spreadsheet.... En nuestro caso, al realizar el prototipo con métodos no profesionales nuestra tarjeta será a doble cara. Por ello indicaremos como capa ruteable tanto la superior (Top) como la inferior (Bottom).

La capa superior se utilizará para los componentes que requieren conexión física, tales como los conectores del PicKit 3, ICD3, así como la conexión con la batería del vehículo, el conector RS232 del Bus Can y el conector al cable plano de la pantalla.

Debido a la complejidad del circuito, especialmente en el ruteo de las pistas del microcontrolador, se realizará un ruteo manual de la mayor parte del circuito. Para ello utilizaremos el icono de Add/Edit Route Mode . Según vayamos creando las pistas iremos bloqueándolas pulsando en la pista y posteriormente con la tecla L. Con ello podremos ir utilizando el autoroute y en el caso de que el ruteo de una determinada zona nos convenza podremos bloquearla. De esta manera al realizar un Unroute/Board sólo se nos eliminarán las pistas con las que no hayamos quedado conformes haciendo así más eficaz el ruteado de la PCB.

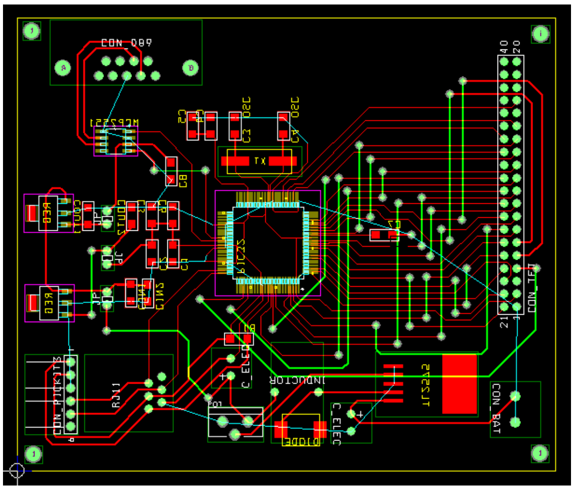


Figura 3.16. Ruteo de pistas. Orcad Layout.

En el diseño de un circuito impreso, y para reducir el efecto del ruido, es aconsejable que las pistas sean lo más cortas posibles siempre que el diseño lo permita. Otra forma de minimizar el ruido es la creación de planos de masa.

Dado que la "masa" o referencia de tensión actúa como camino de retorno tanto para la alimentación como para las señales, siempre que sea posible conviene dedicar al menos una capa del circuito a la tensión de referencia. En nuestro caso, al tratarse de un circuito de doble capa, crearemos unas zonas de masa lo más amplias posibles en cada una de las caras. Esto permitirá que los caminos de retorno sean lo más directos posibles produciéndose el retorno por el camino de menor impedancia. Además, esto facilitará la creación manual de la PCB que será detallada más adelante.

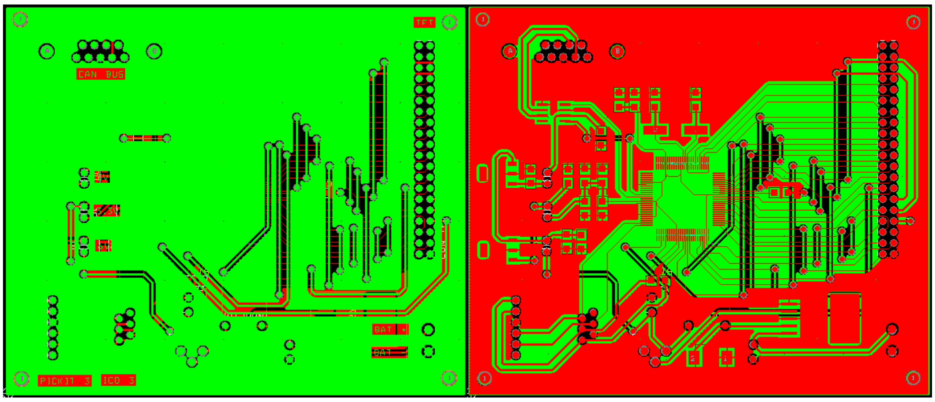


Figura 3.17. Planos de masa. Orcad Layout.

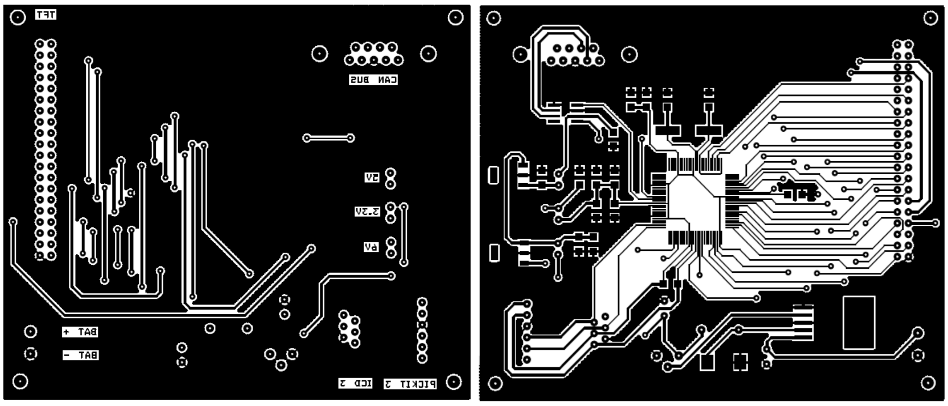


Figura 3.18. Fotolitos. Orcad Layout.

3.3. Fabricación.

La fabricación de la tarjeta de circuito impreso se ha realizado por la técnica de insolado, revelado y atacado por ácido. Se ha utilizado el laboratorio del Departamento de Sistemas Electrónicos y de Control para tal propósito.

Una vez creado diseñada la PCB en Orcad Layout se han impreso los fotolitos de ambas caras en un acetato. Situando los acetatos a ambos lados de un sustrato de FR-4 haciéndolos coincidir. Después se introdujo el sustrato en una insoladora.

Figura 3.19. Insoladora.

El siguiente paso será introducir el sustrato en 3 cubetas diferentes como las que aparecen en la imagen correspondientes al revelado, aclarado y atacado.

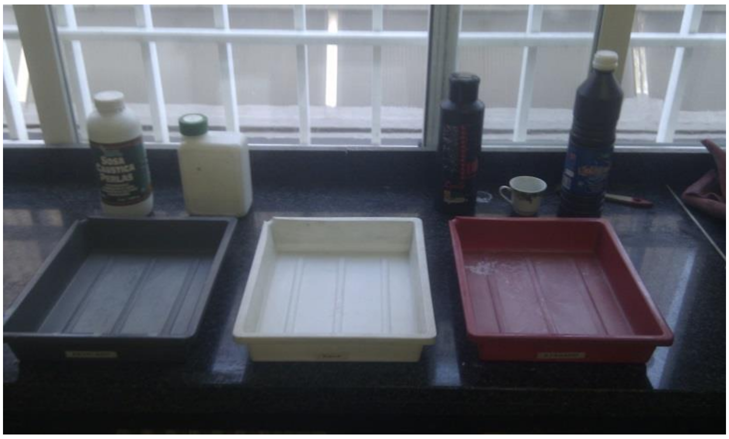


Figura 3.20. Cubetas y líquidos de revelado y atacado.

En la cubeta de revelado se prepara una disolución de sosa cáustica con agua. Dejaremos el sustrato en la cubeta hasta que las pistas sean claramente visibles. Si lo dejáramos demasiado tiempo podrían desaparecer las pistas más críticas del circuito.

La cubeta de aclarado se utiliza para eliminar el material revelador que pueda quedar en la placa.

La última cubeta se prepara con una parte de agua oxigenada 110 volúmenes, otra de ácido clorhídrico. Completamos la mezcla con 2 partes de agua. Introducimos el sustrato y esperamos hasta que la placa esté lista.

Tras completar todo el proceso se taladra la PCB y se realizar la soldadura de los componentes comenzando con el microcontrolador al tratarse del componente más delicado a la hora de la soldadura.

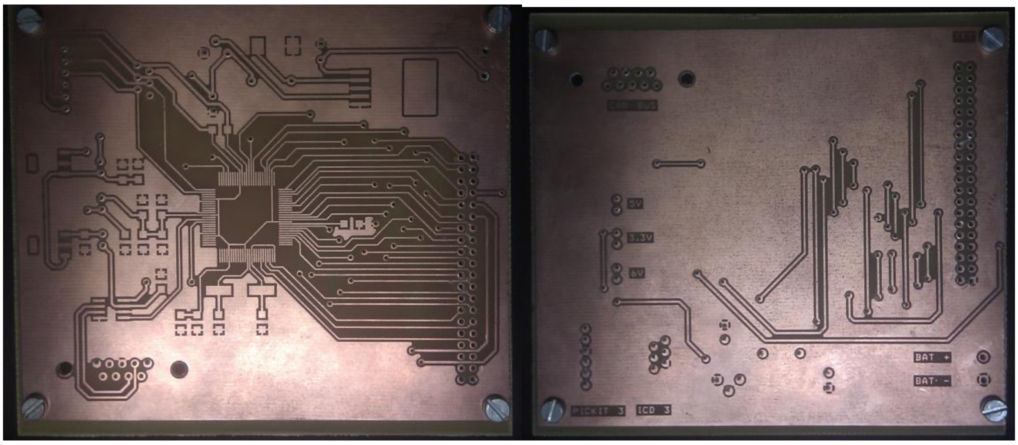


Figura 3.21. PCB previa al soldado de componentes.

3.4. Conexión entre las dos tarjetas (cable plano)

La conexión de la PCB creada está asociada a la configuración propia del módulo de la pantalla. Este módulo contaba en inicio con 40 taladros metalizados para conectar otros módulos.

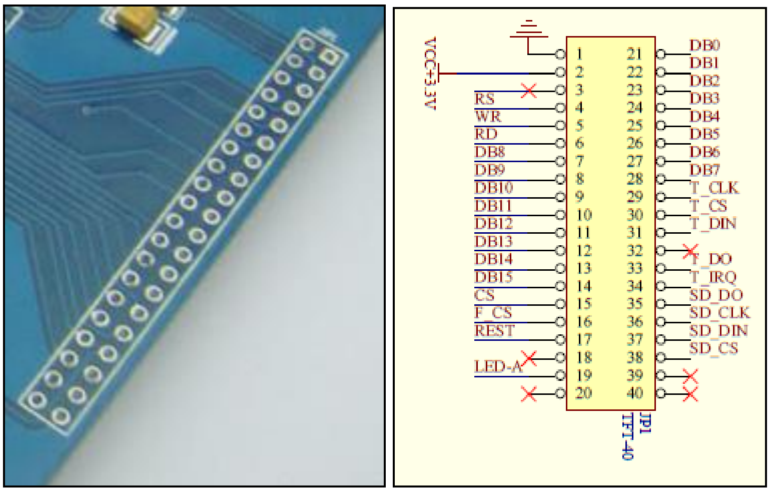


Figura 3.22. Taladros metalizados pantalla táctil.

Contando con que la pantalla traía estos 40 pines se decidió crear en la PCB del microcontrolador otro conexionado de 40 pines para unir las dos placas mediante un cable plano, que permitiese la conexión y desconexión de ambas en caso de ser necesario.

Se adquirió tanto el cable plano como los conectores macho (soldados a las PCBs) y hembra (unidos al cable plano) correspondientes a ambos extremos de unión. El montaje de los conectores del cable plano se realizó en el laboratorio del departamento con ayuda de una mordaza de banco de carpintero disponible en el departamento.

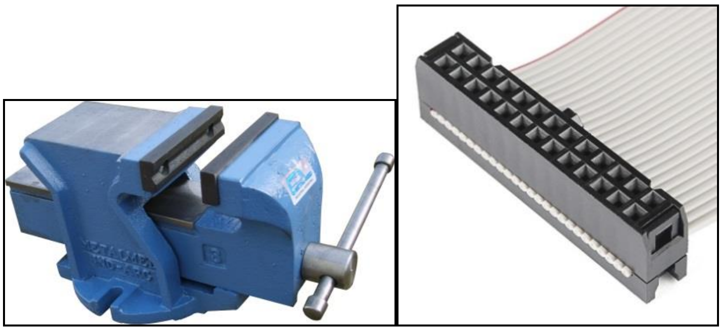


Figura 3.23. Creación de cable plano.

El resultado definitivo de conexión entre ambas placas se puede observar en las siguientes imágenes.

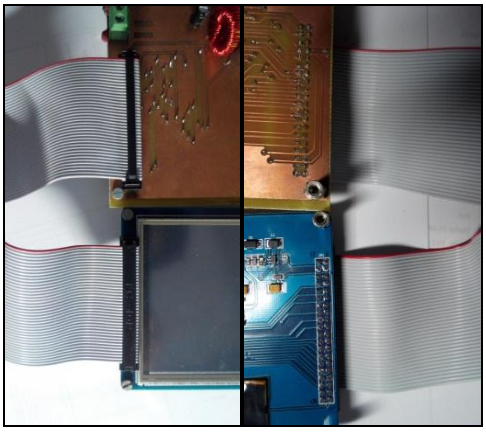


Figura 3.24. Conexión de cable plano.

4. DESARROLLO SOFTWARE

En este capítulo se expondrá todo lo referente al desarrollo software del sistema. Como ya ocurriera en el desarrollo hardware, se explicará cada grupo funcional desde pero desde el punto de vista del software. Para facilitar la comprensión del código de aplicación (disponible tanto en los anexos como en el cd del proyecto) se utilizarán diagramas de estado, esquemas e imágenes tomadas en el laboratorio.

4.1. Entorno de desarrollo MPLABX. Compilador MPLAB XC32.

Dado que nuestro sistema se basa en el microcontrolador PIC32 de la familia Microchip hemos utilizado el entorno de desarrollo MPLABX en su versión v1.95. **MPLAB** es un entorno de desarrollo integrado (IDE) destinado a desarrollar aplicaciones con productos de Microchip. Se trata de un editor modular que permite seleccionar distintos microcontroladores soportados además de permitir la grabación mediante un programador. El entorno se puede descargar gratuitamente desde la página oficial de Microchip. Al trabajar con todos los componentes de Microchip (microcontrolador, IDE y programadores) no existe ningún problema de compatibilidad.

Además del entorno de desarrollo es necesario un **compilador de lenguaje C** encargado de traducir el lenguaje de alto nivel a lenguaje de máquina. Microchip también nos facilita la tarea con una versión gratuita de su compilador para PICs de 32 bits XC32.

Con el compilador y el IDE tenemos todo lo necesario para la programación del PIC y sus periféricos. Sin embargo en capítulos posteriores veremos que necesitamos instalar una **librería gráfica** específica para todo el control gráfico de la pantalla, que será un capítulo explicado con especial detalle debido a que concentra la mayor parte del código del programa.

4.2. Oscilador

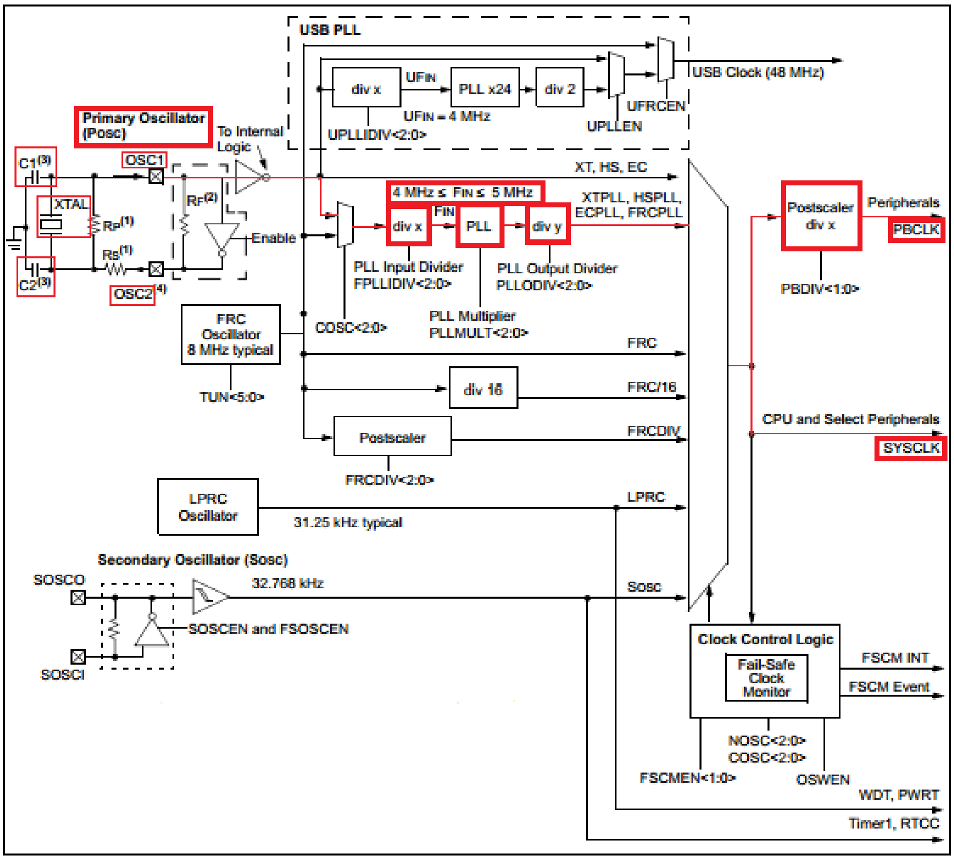


Figura 4.1. Diagrama circuito oscilador.

En la imagen anterior se puede observar el diagrama de bloques completo del sistema oscilador del PIC32. Mediante el recorrido seguido por la línea roja del diagrama se consigue que, a partir del oscilador externo de 8MHz, y mediante el PLL del micro, un reloj del bus principal (SYSCLK) de 80MHz y un reloj del bus de periféricos (PBCLK) de 10MHz.

El primer paso es especificar FPLLIDIV para cumplir con la condición de F_{IN} :

$$4MHz \leq F_{IN} \leq 5MHz$$

$$F_{IN} = \frac{FOSC}{FPLLIDIV} = \frac{8MHz}{2} = 4MHz$$

Una vez cumplida esta primera condición multiplicaremos los 4MHz por 20 mediante PLLMULT:

$$F_{IN} \times PLLMULT = 4MHz \times 20 = 80MHz$$

Con los 80MHz obtenidos sólo tenemos que configurar PLLODIV= 1 y ya tendremos el reloj del sistema deseado:

$$XTPLL = 80MHz \frac{1}{PLLODIV} = 80MHz = SYSCLK$$

$$[\boldsymbol{SYSCLK = 80MHz}]$$

La configuración del bus de periféricos (PBCLK) sólo requiere de la división mediante PBDIV:

$$PBDIV = \frac{XTPLL}{PBDIV} = \frac{80MHz}{8} = 10MHz$$

$$[\boldsymbol{PBDIV = 10MHz}]$$

En cuanto al **código** referente a la configuración del oscilador para la obtención de los relojes previamente calculados, los PIC de 32 bits utilizan unos registros de configuración que comienzan con la sentencia #pragma config.

Utilizamos el Oscilador Primario junto con el PLL:

```
#pragma config FNOSC  = PRIPLL
```

Utilizaremos un cristal externo de 8MHz:

```
#pragma config POSCMOD  = XT
```

Divisor de entrada PLL:

```
#pragma config FPLLIDIV = DIV_2
```

Multiplicador PLL:

```
#pragma config FPLLMUL  = MUL_20
```

Divisor de salida:

```
#pragma config FPLLODIV = DIV_1
```

Divisor del bus de periféricos:

```
#pragma config FPBDIV   = DIV_1
```

4.3. PROGRAMACIÓN MEDIANTE ICD3 Y PICKit3.

La única configuración necesaria en el bloque de programación es dependiente del hardware elegido, y más en concreto, de los pines con los que vamos a programar. En nuestro caso utilizaremos los pares de pines PGED2/PGEC2 y lo indicaremos al PIC mediante la siguiente sentencia de programación.

```
#pragma config ICESEL   = ICS_PGx2
```

4.4. CAN BUS

4.4.1. Requisitos previos. Velocidad de transmisión. Tiempo de bit.

Como **requisito previo** a la configuración propiamente dicha del CAN, y como se indicó anteriormente en el apartado del hardware, al módulo CAN se puede acceder desde diferentes pines (C1RX y C1TX y los pines alternativos AC1RX AC1TX).

Con la siguiente sentencia de configuración del PIC indicaremos que vamos a utilizar los pines alternativos CAN.

```
#pragma config FCANIO    = OFF
```

Para la **configuración CAN** se utilizan los registros y funciones proporcionados por el fabricante Microchip. Las funciones se encuentran dentro de CAN.h y los registros a configurar son divididos según Microchip en 4 grupos (datasheet Section 34: Controller Area Network (CAN)):

- Registros de configuración general y de velocidad de transmisión.
- Registros de interrupción y de estado.
- Registros de configuración de filtros y máscaras.
- Registros del módulo de control CAN.

La **velocidad de transmisión** del bus es el número de bits por segundo transmitidos en el bus. Nuestro módulo CAN nos permite programar una velocidad de hasta 1Mbps. En nuestro caso configuraremos nuestro sistema a una velocidad de 125kbps, más que suficiente para nuestra aplicación.

Así mismo, la velocidad del can bus viene ligada a la longitud máxima del cableado. Como veremos en la tabla a continuación que relaciona la velocidad y la longitud del bus. Como es de suponer, la velocidad es inversamente proporcional a la longitud que nos permitiría el sistema CAN Bus.

Velocidad [Kbps]	Longitud [metros]
1000	40
500	100
250	200
125	500
50	1000
5	10000

Figura 4.2. Relación velocidad-longitud CAN Bus.

Nuestro sistema se ha creado para funcionar sobre un vehículo, por tanto no tenemos ningún problema en cuanto a distancia del cableado, pudiendo aumentar la velocidad hasta 1Mbps en caso de que algún nodo nuevo situado en el coche lo requiriera.

```
#define CAN_BUS_SPEED  (125000)
```

El tiempo de bit o **Nominal Bit Time** representa la duración de un bit en el bus. Cada nodo que se conecte al bus debe estar configurado con el mismo tiempo de bit para poder emitir y recibir correctamente los mensajes.

$$\left[Nominal\ Bit\ Time = \frac{1}{Nominal\ Bit\ Rate} = \frac{1}{125kbps} = 8\mu S\right]$$

El tiempo de bit está formado por 4 segmentos de duración determinada (T_Q) y configurable por el usuario siempre que se encuentre entre 8 y 25 T_Q en total. En nuestro caso y debido a que nos conectaremos al simulador de Can Bus utilizaremos los mismos parámetros de tiempo de bit que vienen configurados en el mismo por defecto. Será necesario definir estos parámetros en la función de inicialización CAN1Init().

Syncronization Segment = $1\ T_Q$

Propagation Segment =$3\ T_Q$

Phase Segment 1 = $3\ T_Q$

Phase Segment 2 = $3\ T_Q$

$$\left[T_Q\ por\ bit = 1T_Q + 3T_Q + 3T_Q + 3T_Q = 10T_Q\right]$$

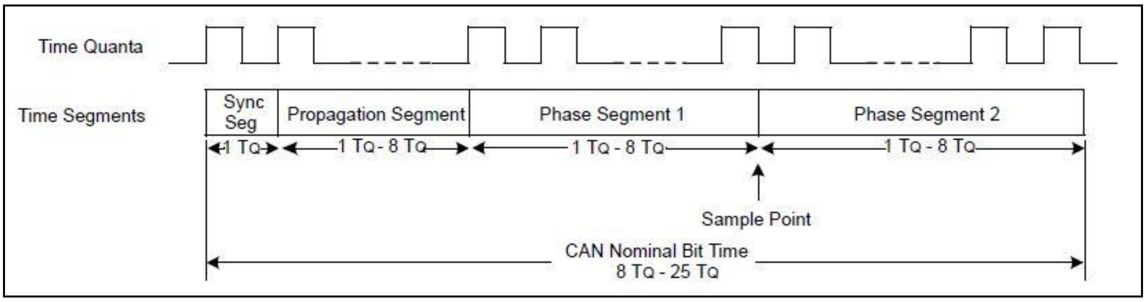


Figura 4.3. Tiempo de bit.

4.4.2. Funciones principales.

4.4.2.1. Configuración inicial.

Es la función principal de configuración del periférico. CAN1Init() configurará los parámetros generales del bus como la velocidad de transmisión y el tiempo de bit. Además establecerá las características (tamaño, filtros, interrupciones, eventos…) de los canales de transmisión y recepción.

Una vez realizado todo el proceso de configuración dejará el CAN bus en modo NORMAL y preparado para enviar y recibir mensajes que se gestionarán en sus respectivas funciones.

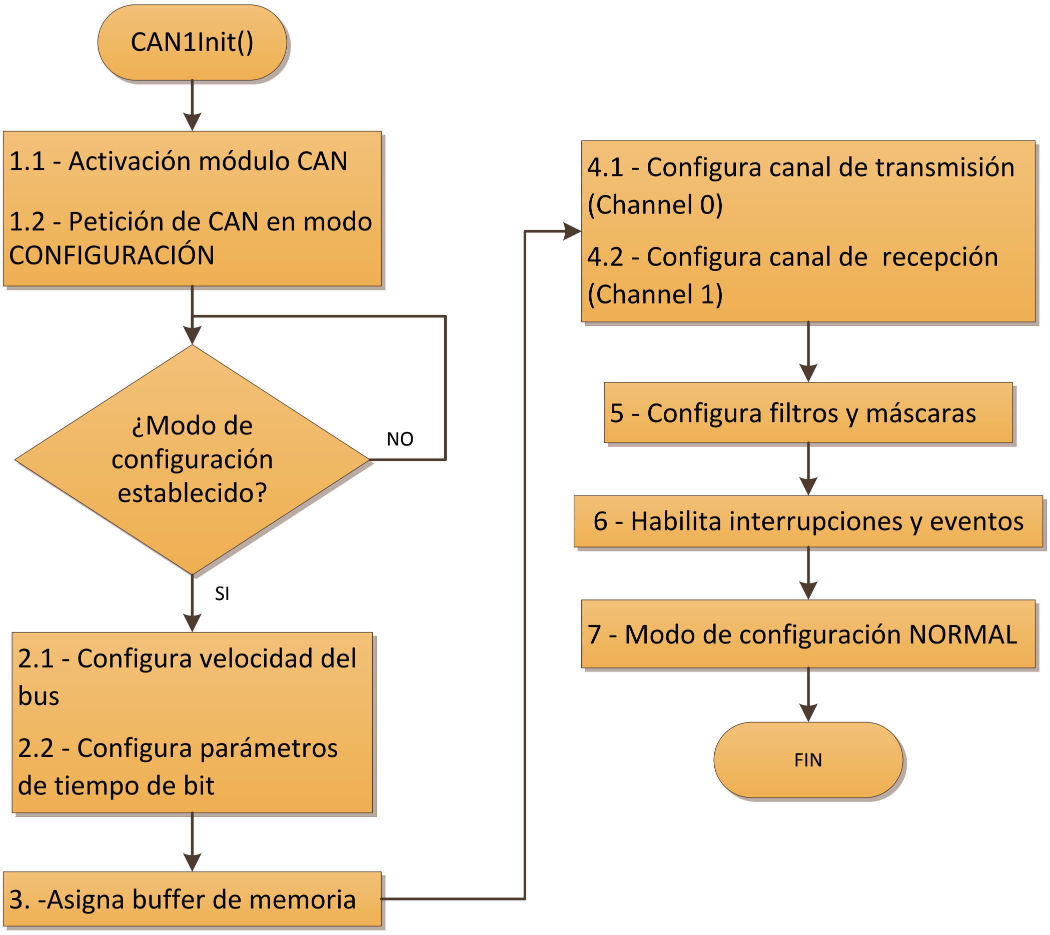


Figura 4.4. Diagrama de estados. CAN1Init.

4.4.2.2. Envío CAN Bus.

Con la función CAN1Transmit seremos capaces de transmitir una trama de datos mediante el canal 0 (canal creado para la transmisión) en el bus.

Figura 4.5. Diagrama de estados. CAN1Transmit.

4.4.2.3. Recepción CAN Bus.

La recepción de tramas del sistema se realiza a través de interrupciones del PIC. El PIC32 genera peticiones de interrupción en respuesta a eventos de sus periféricos. En la inicialización (CAN1Init) se configuraron y habilitaron los eventos de pila no vacía y el evento de recepción. También se habilitaron las interrupciones propias del periférico. Por tanto, una vez que se produzca la interrupción se procederá a almacenar y a procesar la trama.

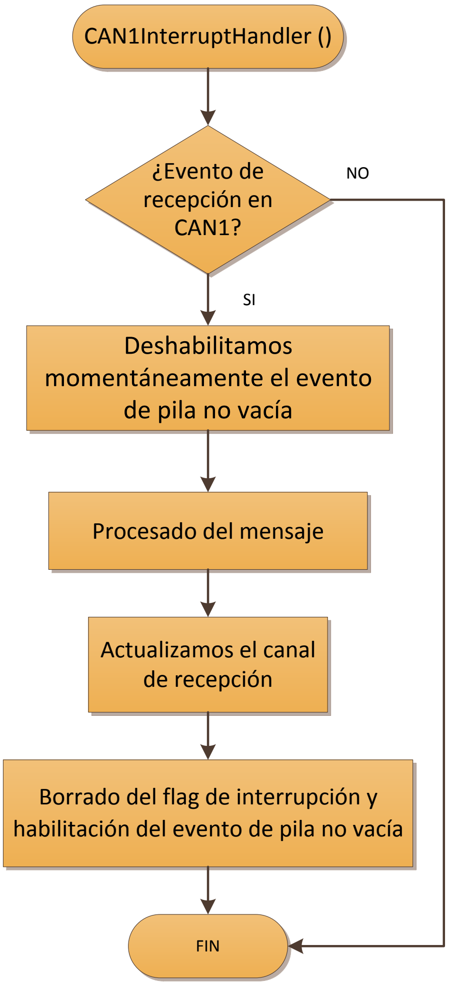


Figura 4.6. Diagrama de estados. CAN1InterruptHandler.

4.4.3. Entorno de pruebas. SIMULADOR.

Para verificar el correcto funcionamiento del CAN Bus en nuestro sistema y debido a que no seremos capaces de probar el proyecto dentro del coche junto a los sensores propios de éste, se ha utilizado una tarjeta simuladora de CAN Bus. La tarjeta se conoce como MCP2515 CAN Bus Monitor y ha sido proporcionada por el departamento.

4.4.3.1. MCP2515 CAN Bus Monitor.

El MCP2515 CAN Bus Monitor es una herramienta creada por Microchip para poder testear el protocolo CAN con un bajo coste. En nuestro caso lo utilizaremos para simular los sensores propios del vehículo mediante el envío y la recepción de tramas.

El contenido del kit es el siguiente :

- 2 tarjetas idénticas que harán de nodos del sistema.
- Un cable genérico USB a mini-USB para conectar una de las tarjetas al pc, alimentar al sistema e intercambiar información entre en software y el sistema CAN.
- Un cable CAN con 3 puntos de acceso. A los extremos para conectar con las tarjetas de desarrollo y entre ellos un conector RS232. Este último nos permitirá introducir un nuevo nodo (nuestro sistema) para realizarle el teste oportuno.
- MCP2515DM-BM PC Software Rev2.0 (MCP2515 Bus Monitor). Software principal del programa que instalaremos en nuestro pc y desde el cual monitorizaremos las tramas del protocolo CAN.

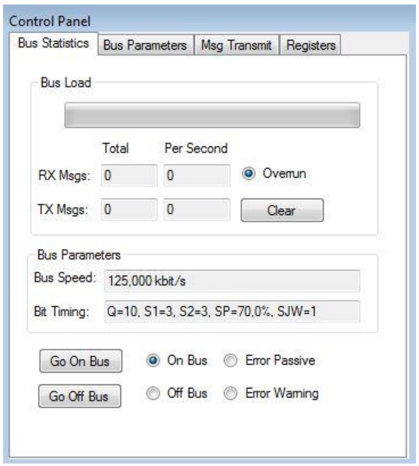
El kit de desarrollo consiste en dos tarjetas idénticas que, mediante la correcta conexión entre ellas a través del cable CAN y con el pc, nos permiten crear un sistema CAN de 2 nodos.

Los dos nodos son idénticos, sin embargo, la función de cada uno de ellos viene determinada por la conexión del sistema completo. Uno de los nodos se convertirá en el monitor (el nodo conectado al PC mediante el USB) y el otro será el generador de tráfico. También será posible utilizar un único nodo que sirva tanto de monitor como de generador de tráfico.

4.4.3.2. Funcionamiento del programa.

El software asociado al simulador cuenta con 4 pestañas y una ventana de visualización de tramas.

- Estadísticas del bus: muestra las estadísticas más relevantes del bus, tales como la carga de tráfico que está soportando, la velocidad del bus o el número de mensajes transmitidos y recibidos, entre otros.



	Total	Per Second
RX Msgs:	0	0
TX Msgs:	0	0

● Overrun

Bus Parameters

Bus Speed: 125,000 kbit/s

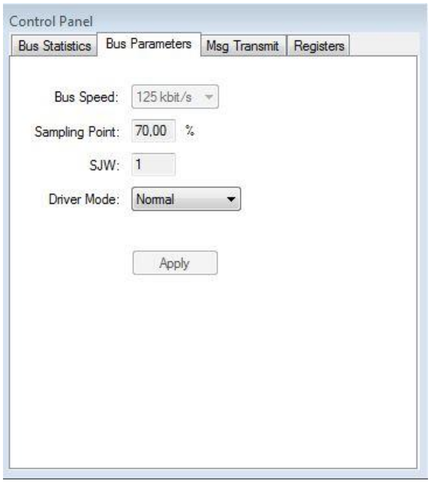
Bit Timing: Q=10, S1=3, S2=3, SP=70,0%, SJW=1

● On Bus ○ Error Passive

○ Off Bus ○ Error Warning

Figura 4.7. Pestaña de estadísticas del bus.

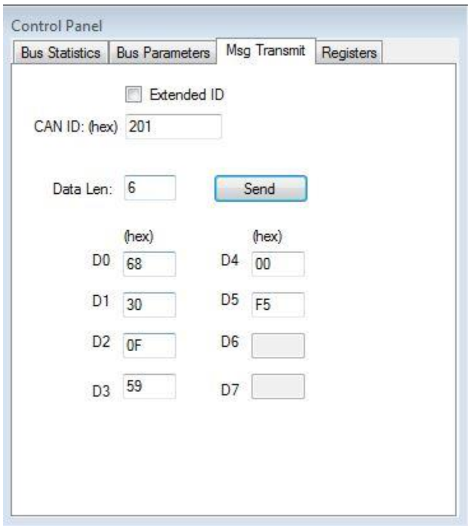
- Configuración de parámetros: en esta ventana se configurarán los parámetros del bus y se seleccionará el estado del simulador.



Bus Speed: 125 kbit/s
Sampling Point: 70,00 %
SJW: 1
Driver Mode: Normal

Figura 4.8. Pestaña de configuración de parámetros.

- Ventana de transmisión de mensajes: desde esta ventana crearemos las tramas que deseemos enviar a través del bus. Elegiremos el ID de la trama, si se trata o no de una trama extendida, el número de datos y los datos a enviar.

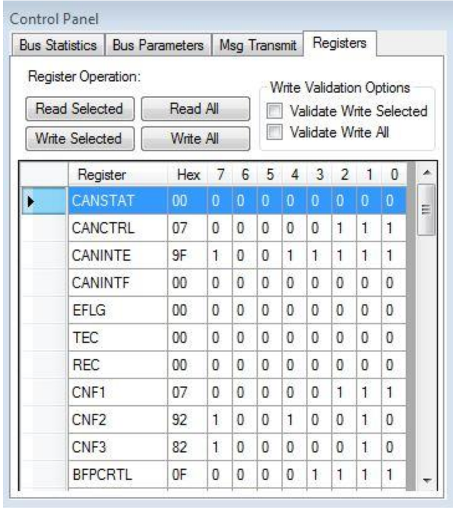


☐ Extended ID
CAN ID: (hex) 201
Data Len: 6

(hex)		(hex)	
D0	68	D4	00
D1	30	D5	F5
D2	0F	D6	
D3	59	D7	

Figura 4.9. Pestaña de transmisión de tramas.

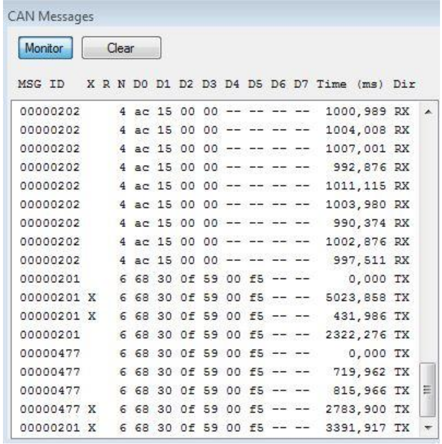
- Registros : aquí se muestran los registros asociados al MCP2515 que contiene la tarjeta simuladora.



Register	Hex	7	6	5	4	3	2	1	0
CANSTAT	00	0	0	0	0	0	0	0	0
CANCTRL	07	0	0	0	0	0	1	1	1
CANINTE	9F	1	0	0	1	1	1	1	1
CANINTF	00	0	0	0	0	0	0	0	0
EFLG	00	0	0	0	0	0	0	0	0
TEC	00	0	0	0	0	0	0	0	0
REC	00	0	0	0	0	0	0	0	0
CNF1	07	0	0	0	0	0	1	1	1
CNF2	92	1	0	0	1	0	0	1	0
CNF3	82	1	0	0	0	0	0	1	0
BFPCRTL	0F	0	0	0	0	1	1	1	1

Figura 4.10. Pestaña de registros.

- Mensajes CAN: En la parte derecha veremos un módulo asociado a esta ventana que muestra todos los mensajes que están atravesando el bus. Aquí veremos los mensajes que enviemos desde el simulador (TX) al sistema y los enviados desde el sistema al bus (RX).



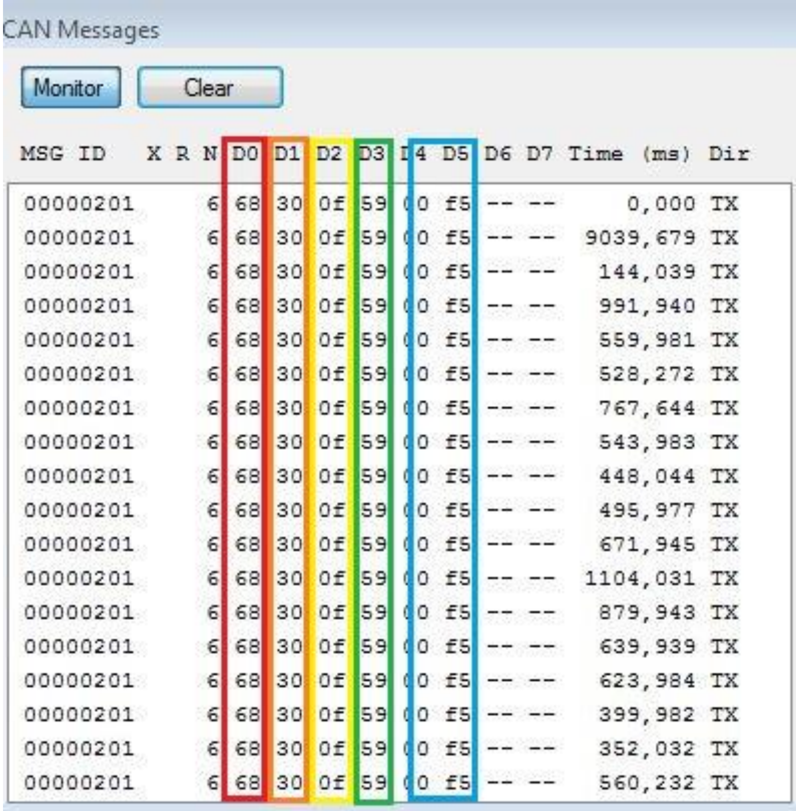
```
MSG ID    X R N D0 D1 D2 D3 D4 D5 D6 D7 Time (ms) Dir
00000202      4 ac 15 00 00 -- -- -- --  1000,989 RX
00000202      4 ac 15 00 00 -- -- -- --  1004,008 RX
00000202      4 ac 15 00 00 -- -- -- --  1007,001 RX
00000202      4 ac 15 00 00 -- -- -- --   992,876 RX
00000202      4 ac 15 00 00 -- -- -- --  1011,115 RX
00000202      4 ac 15 00 00 -- -- -- --  1003,980 RX
00000202      4 ac 15 00 00 -- -- -- --   990,374 RX
00000202      4 ac 15 00 00 -- -- -- --  1002,876 RX
00000202      4 ac 15 00 00 -- -- -- --   997,511 RX
00000201      6 68 30 0f 59 00 f5 -- --     0,000 TX
00000201 X    6 68 30 0f 59 00 f5 -- --  5023,858 TX
00000201 X    6 68 30 0f 59 00 f5 -- --   431,986 TX
00000201      6 68 30 0f 59 00 f5 -- --  2322,276 TX
00000477      6 68 30 0f 59 00 f5 -- --     0,000 TX
00000477      6 68 30 0f 59 00 f5 -- --   719,962 TX
00000477      6 68 30 0f 59 00 f5 -- --   815,966 TX
00000477 X    6 68 30 0f 59 00 f5 -- --  2783,900 TX
00000201 X    6 68 30 0f 59 00 f5 -- --  3391,917 TX
```

Figura 4.11. Ventana de mensajes CAN.

4.4.3.3. Trama simulada de los sensores del vehículo.

Dado que nuestro sistema no se ha podido probar en el vehículo, para probar el correcto funcionamiento del programa creamos tramas definidas para enviar a nuestro sistema. Las tramas se crearán desde la ventana de transmisión de mensajes y tendrán la siguiente estructura.

- D0. RPM x 1000: el byte D0 formado por dos dígitos hexadecimales contendrá la información de las revoluciones por minuto x 1000 del motor.
- D1. Temperatura: el byte D1 formado por dos dígitos hexadecimales contendrá la información de la temperatura del agua suponiéndose 100% la temperatura crítica para el funcionamiento del vehículo y 80% una temperatura que requerirá cambiar algún parámetro o tener especial precaución en la conducción.
- D2. Temperatura del aceite: el byte D2 formado por dos dígitos hexadecimales contendrá la información de la temperatura del aceite. Al igual que sucedía con la temperatura del agua, se supondrá 100% un nivel crítico y un 80% un nivel elevado de temperatura.
- D3. Batería: el byte D3 formado por dos dígitos hexadecimales contendrá la información de la carga de la batería en %. Al contrario de las temperaturas, el nivel de emergencia estará entre 0 y 20%.
- D4 y D5. Velocidad: Los 3 dígitos hexadecimales que forman el byte de D5 y un dígito hexadecimal de D4 contendrán la información de la velocidad actual del vehículo.



```
MSG ID     X R N D0 D1 D2 D3 D4 D5 D6 D7 Time  (ms) Dir
00000201         6 68 30 0f 59 00 f5 -- --     0,000 TX
00000201         6 68 30 0f 59 00 f5 -- --  9039,679 TX
00000201         6 68 30 0f 59 00 f5 -- --   144,039 TX
00000201         6 68 30 0f 59 00 f5 -- --   991,940 TX
00000201         6 68 30 0f 59 00 f5 -- --   559,981 TX
00000201         6 68 30 0f 59 00 f5 -- --   528,272 TX
00000201         6 68 30 0f 59 00 f5 -- --   767,644 TX
00000201         6 68 30 0f 59 00 f5 -- --   543,983 TX
00000201         6 68 30 0f 59 00 f5 -- --   448,044 TX
00000201         6 68 30 0f 59 00 f5 -- --   495,977 TX
00000201         6 68 30 0f 59 00 f5 -- --   671,945 TX
00000201         6 68 30 0f 59 00 f5 -- --  1104,031 TX
00000201         6 68 30 0f 59 00 f5 -- --   879,943 TX
00000201         6 68 30 0f 59 00 f5 -- --   639,939 TX
00000201         6 68 30 0f 59 00 f5 -- --   623,984 TX
00000201         6 68 30 0f 59 00 f5 -- --   399,982 TX
00000201         6 68 30 0f 59 00 f5 -- --   352,032 TX
00000201         6 68 30 0f 59 00 f5 -- --   560,232 TX
```

Figura 4.12. Estructura de la trama simulada.

4.5. Control táctil.

4.5.1. Estudio del panel táctil.

El primer paso en el estudio del panel táctil fue comprobar que datos nos entregaba el controlador táctil en cada lugar de la pantalla. Se realizaron medidas en puntos concretos del panel con un lápiz táctil consiguiendo así mayor precisión en las medidas. Las medidas se realizaron en puntos concretos como las esquinas y el centro del panel.

El controlador entrega 3700 datos en cada eje, comenzando en 200 el valor mínimo y con 3900 como valor máximo. Sin embargo, y debido a la calidad del mismo existe un ruido interno que se comprueba al tomar varias medidas en el mismo punto con el lápiz táctil. Obviamente el ruido producido al pulsar con un dedo es mayor.

Esta ineficacia en las medidas no es alarmante debido a que contamos con una pantalla de unas dimensiones más que suficientes posibilitándonos la creación de botones de un tamaño apropiado para que el usuario apenas note este ruido.

Al fijarnos en el origen de coordenadas de los datos entregados por el controlador observamos que la correspondencia con las coordenadas de la pantalla con las que trabajaremos es contraria. Las coordenadas por tanto las obtendremos con los siguientes cálculos.

$$K_x = \frac{3900 - 200}{800} = 4{,}625$$

$$K_x = \frac{3900 - 200}{480} = 7{,}708$$

$$\left[\boldsymbol{Coordenada\ X = 800 - (\frac{Medida_X - 200}{4{,}625})}\right]$$

$$Coordenada\ Xmín = 800 - \left(\frac{Medida_x máx - 200}{4{,}625}\right) =$$

$$= 800 - \left(\frac{3900 - 200}{4{,}625}\right) = 0$$

$$Coordenada\ Xmáx = 800 - \left(\frac{Medida_X mín - 200}{4{,}625}\right) =$$

$$= 800 - \left(\frac{200 - 200}{4{,}625}\right) = 800$$

$$\left[\boldsymbol{Coordenada\ Y = 480 - \left(\frac{Medida_Y - 200}{7{,}708}\right)}\right]$$

$$Coordenada\ Ymín = 480 - \left(\frac{Medida_Y máx - 200}{7{,}708}\right) =$$

$$= 480 - \left(\frac{3900 - 200}{7{,}708}\right) = 0$$

$$Coordenada\ Ymáx = 480 - \left(\frac{Medida_X mín - 200}{7{,}708}\right) =$$

$$= 480 - \left(\frac{200 - 200}{7{,}708}\right) = 480$$

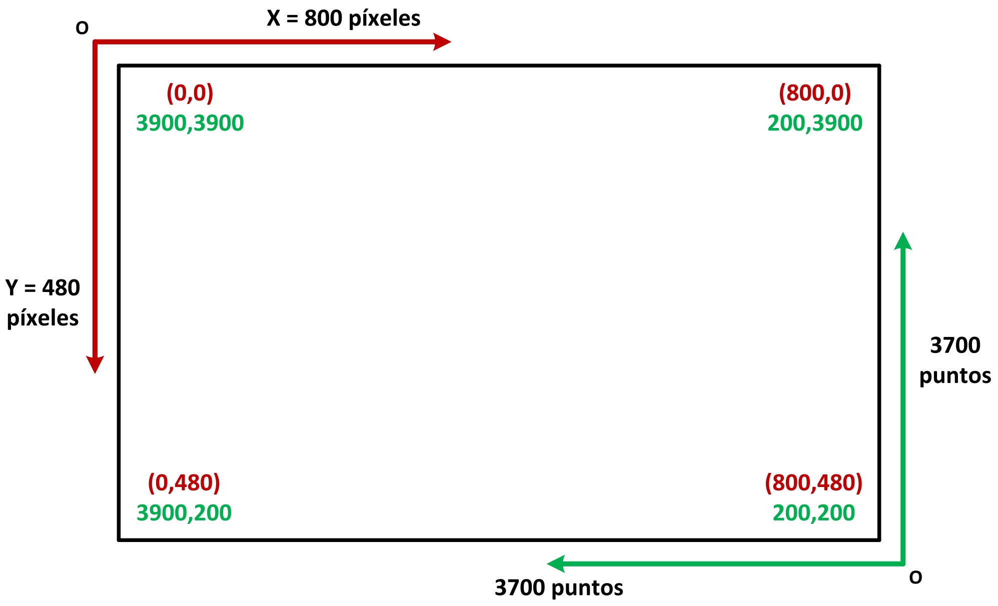


Figura 4.13. Relación coordenadas y datos a la salida del controlador táctil.

4.5.2. Función de lectura de coordenadas.

La función principal de lectura de coordenadas se sitúa dentro del bucle de actualización de imágenes. Las coordenadas obtenidas son comparadas con las coordenadas de la lectura anterior. De esta forma somos capaces de establecer que evento ha ocurrido y actuar en consecuencia instantáneamente. Los eventos posibles son los siguientes:

- EVENT_INVALID : en el caso de una pulsación fuera de los límites del panel, tan cerca del límite que obtengamos valores no deseables o en el caso en que no se realice ninguna pulsación.
- EVENT_PRESS: la primera lectura de una pulsación siempre obtendrá como resultado un EVENT_PRESS o en su defecto un EVENT_INVALID dependiendo si estamos comparando con una lectura válida anterior o no.

- EVENT_STILLPRESS: en el caso de que la lectura previa de pulsación y la actual tengan el mismo valor.

- EVENT_MOVE: en el caso de que la pulsación actual difiera del valor anterior, sin dejar de pulsar la pantalla. Se trata de un “deslizamiento”, por ejemplo utilizado para mover un widget de desplazamiento como el utilizado en el control de brillo. También permite que desplacemos el dedo dentro del mismo botón esperando que se deje de pulsar.

- EVENT_RELEASE: en la lectura del controlador ha dejado de existir un dato, por tanto se ha dejado de pulsar en el panel táctil. Se diferencia de EVENT_INVALID en que en la lectura anterior existía un dato válido del tipo EVENT_MOVE o EVENT_STILLPRESS.

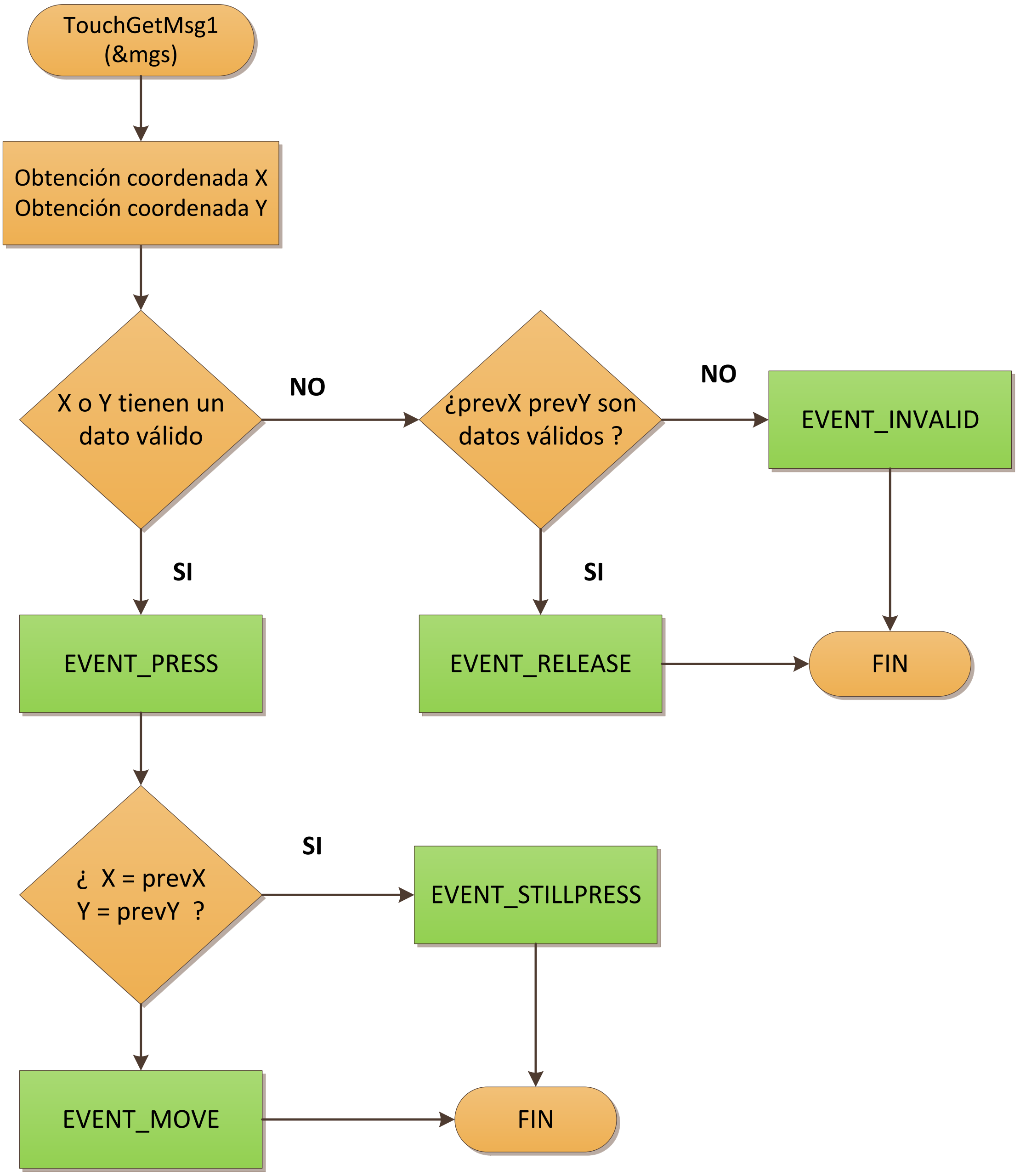


Figura 4.14. Diagrama de estados. TouchGetMsg1.

4.6. Control grafico

El control gráfico se puede considerar otro bloque funcional del sistema. Sin embargo y debido al tipo de proyecto que se ha desarrollado, concentra la mayor parte del software y por tanto del tiempo invertido en su desarrollo.

Como se comentó en el capítulo de hardware de este mismo bloque, se necesita además de un microcontrolador (PIC32MX795F512L) un controlador gráfico (SSD1963) para el procesado de imágenes. Además en el desarrollo software y dado que se va a programar una pantalla necesitamos una **librería gráfica** específica.

4.6.1. Configuración del PMP.

El Puerto Paralelo Maestro se encarga de la comunicación entre el microcontrolador y el PIC. Los PICs de 32 bits tienen grandes opciones de configuración del Puerto Paralelo Maestro.

- Modo de configuración :

  Seleccionaremos el Master Mode 2 que activa los pines PMCSx, PMRD, PMWR, PMD [0:15] que corresponden a los pines conectados físicamente con el SSD1963.

  ```
  PMMODEbits.MODE   = 2;
  ```

- Ancho del bus de datos:

  El PMP soporta tanto 8 como 16 bits de ancho de datos. En nuestro caso seleccionaremos el modo de16 bits.

  ```
  PMMODEbits.MODE16 = 1;
  ```

- Chip Select:

  Utilizaremos el Chip Select CS1.

PMCONbits.CSF = 2;

- Estados de espera:

  En el modo maestro, el usuario puede controlar la duración de la lectura y la escritura configurando los estados de espera. Un estado de espera corresponde a un ciclo de reloj del bus de periféricos, T_{PBCLK}.

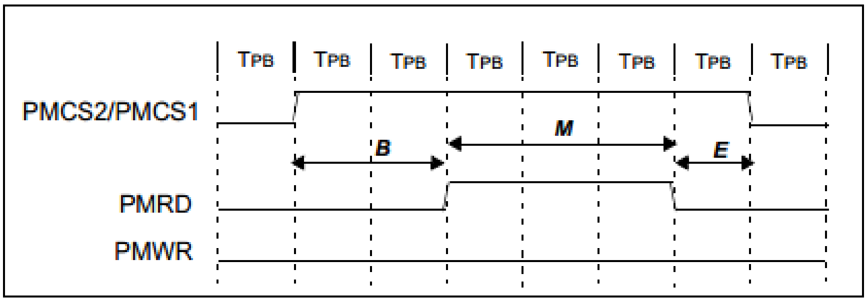


Figura 4.15. Estados de espera en un ciclo de lectura.

4.6.2. Configuración del SSD1963.

El controlador SSD1963 con el que se ha trabajado no cuenta con drivers propios dentro de las librerías gráficas de Microchip, por lo que se han tenido que crear los drivers propios del mismo.

La **configuración del SSD1963** es dependiente del hardware asociado, en nuestro caso de la pantalla conectada. Se realiza en la función ResetDevice() a continuación de la configuración del PMP.

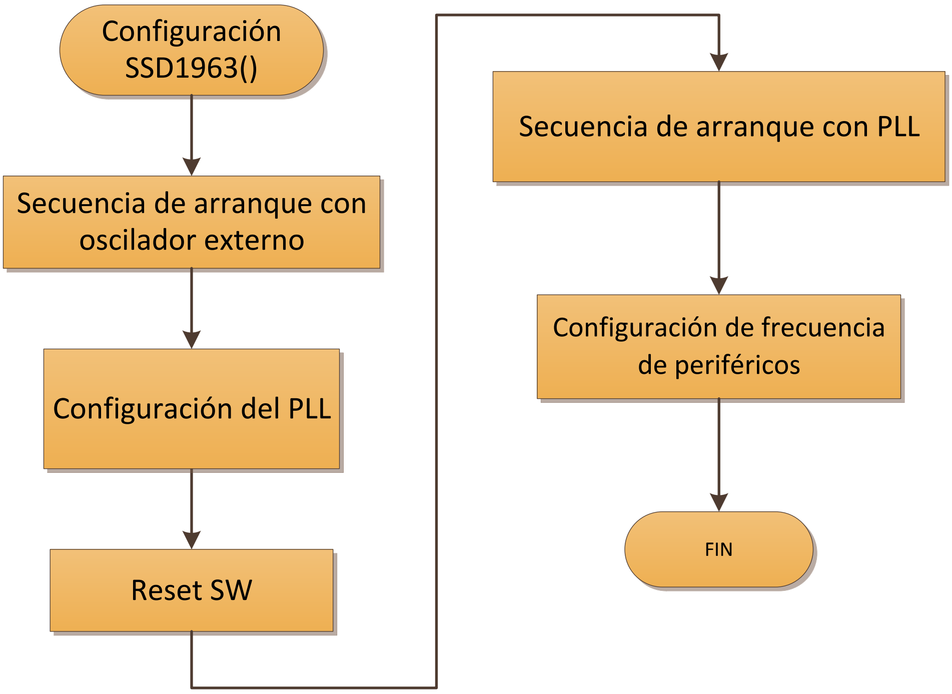


Figura 4.16. Diagrama de estados. Configuración SSD1963.

Tras configurarse los relojes del SSD1963 se procede a **establecer los parámetros de la pantalla**.

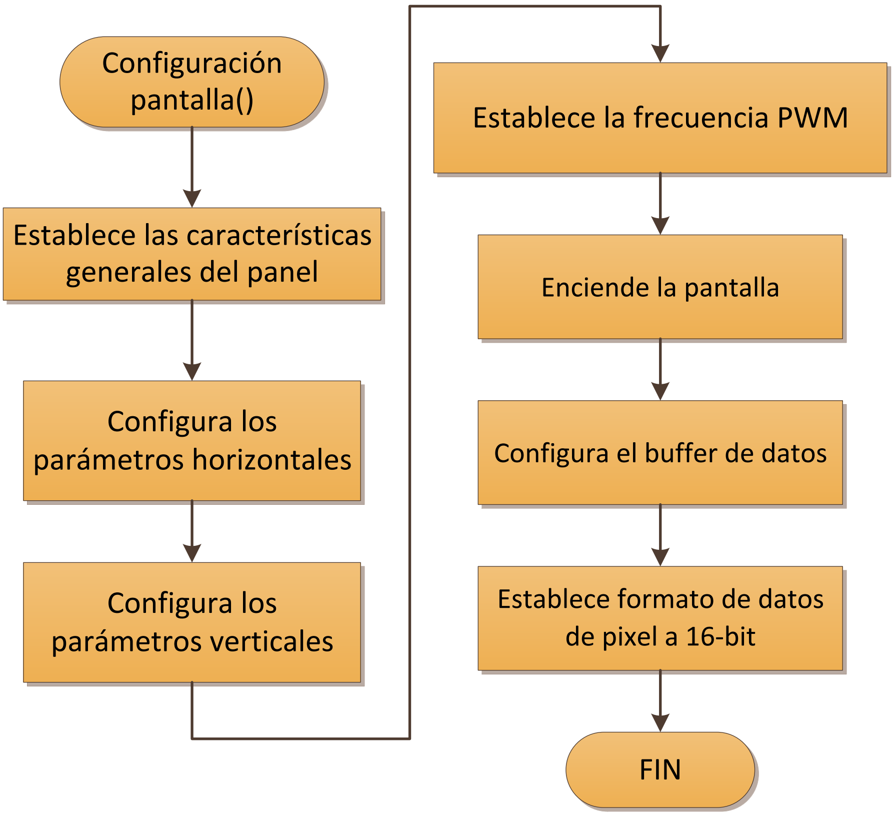


Figura 4.17. Diagrama de estados. Configuración de la pantalla.

La inicialización completa de la pantalla y el controlador seguirá por tanto los siguientes pasos:

1. Antes de la inicialización del dispositivo mediante su Chip Select, se debe asegurar que el Reset se ha mantenido al menos 100µs activo a nivel bajo.

2. Frecuencia de PLL. Para obtener 100MHz:

   Según las ecuaciones:

   $$VCO = Reloj\ de\ entrada \times (M + 1)$$

   $$PLL\ frec = \frac{VCO}{(N + 1)}$$

Siendo 250MHz < VCO < 800MHz

Seleccionando M=29 y N=2 cumplimos tanto el requisito de VCO como la frecuencia de PLL.

$VCO = 10MHz \times (29 + 1) = 300MHz$

$$\left[\boldsymbol{PLL\ frec = \frac{300MHz}{(2+1)} = 100MHz}\right]$$

3. Activación de PLL.

4. Espera de 100µs para la estabilización del PLL.

5. Iniciación de frecuencia del dispositivo ahora con el PLL activo.

6. Reset por software para establecer la nueva frecuencia.

7. Configuración de la frecuencia necesaria para los periféricos, en nuestro caso de la pantalla.

   Según el datasheet de la pantalla necesitamos una DCLK de 33MHz.

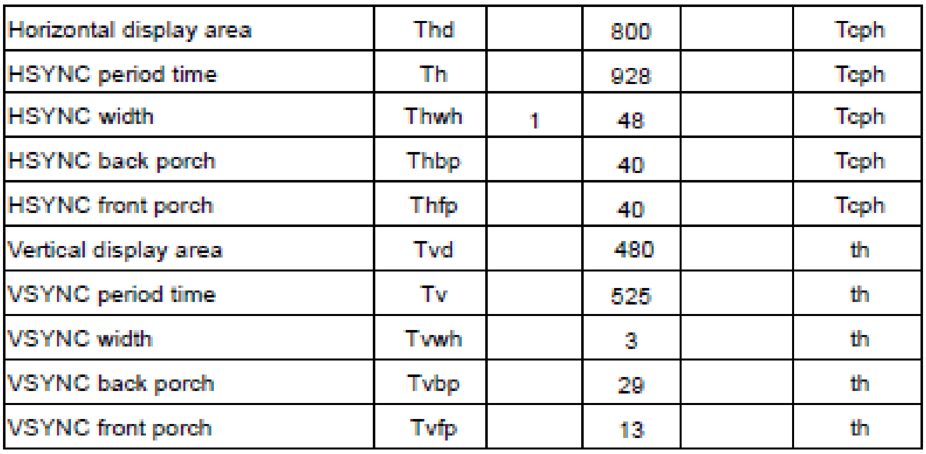
$$33MHz = \frac{100MHz \times (LCDR\_FPR + 1)}{2^{20}}$$

$$LCDR_{FPR} = 346029(0x547AD)$$

A continuación continuamos con la configuración de la pantalla correspondiente al segundo diagrama de estados.

8. Establecemos las características propias de la pantalla que aparecen en su datasheet: 24 bits de datos, FRC deshabilitado, modo TFT, tamaño del panel 800 x 480.

9. Configuración de las características de sincronización vertical y horizontal de la pantalla que aparecen en su datasheet.



Horizontal display area	Thd		800		Tcph
HSYNC period time	Th		928		Tcph
HSYNC width	Thwh	1	48		Tcph
HSYNC back porch	Thbp		40		Tcph
HSYNC front porch	Thfp		40		Tcph
Vertical display area	Tvd		480		th
VSYNC period time	Tv		525		th
VSYNC width	Tvwh		3		th
VSYNC back porch	Tvbp		29		th
VSYNC front porch	Tvfp		13		th

Figura 4.18. Parámetros de la pantalla (datasheet).

10. Configuración de frecuencia PWM.

11. Encendido inicial de la pantalla.

12. Configuración del buffer de datos según el tamaño de la pantalla.

13. La profundidad de color será de 16bits en formato **RGB565**. El RGB es un formato de color basado en la síntesis aditiva, con lo que es posible representar cualquier color por adición de los tres colores primarios. En el caso del RGB565 los colores rojo y azul cuentan con 5 bits cada uno mientras que el color verde está formado por 6 bits. Por tanto la diversidad de colores es 2^{16}=65536 colores.

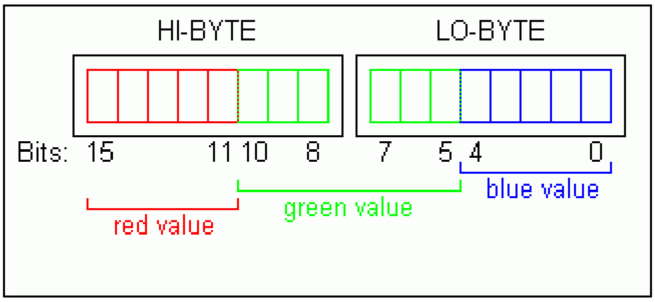


Figura 4.19. RGB565.

4.6.3. Librerías gráficas de Microchip.

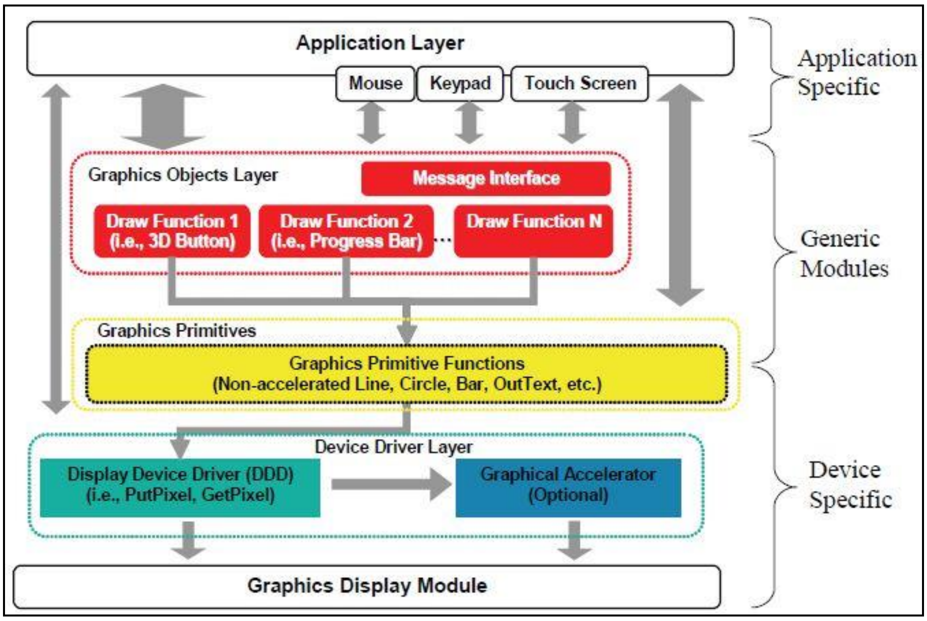


Figura 4.20. Librería gráfica de Microchip.

Microchip también nos proporciona una librería gráfica gratuita así como ejemplos y drivers para diferentes controladores.

Las capas que forman la librería gráfica de Microchip son las siguientes:

- Graphics Display Module: Es la capa formada por los componentes físicos del sistema. En nuestro caso, la pantalla que mostrará las imágenes procesadas.

- Device Driver Layer: En esta capa provee de las funciones más básicas a las capas superiores. Esta capa es dependiente del controlador. Microchip proporciona los drivers correspondientes a esta capa para diferentes controladores. En el caso de utilizar un controlador diferente, como es nuestro caso, se deberán crear los drivers específicos.

  Una de las funciones más importantes que proporciona esta capa es PutPixel(x, y). PutPixel "dibuja" un pixel en la coordenada x e y de la pantalla.

- Graphics Primitives: A partir de las funciones básicas de la capa anterior, esta capa proporciona las funciones necesarias para crear líneas, rectángulos, círculos, texto, etc.

- Graphics Objects Layers: En esta capa se proporcionan las funciones necesarias para la creación de **widgets**. Los widgets son objetos con una funcionalidad específica dentro del sistema gráfico. Microchip proporciona desde widgets sencillos como botones de todas formas y tamaños, hasta widgets más específicos como barras de progreso, potenciómetros, cajas de texto, etc. Cuando se dibuja un en la pantalla sigue un estilo que determina por ejemplo el color del objeto, el tipo de fuente que utiliza, tamaño, color etc. Podemos crear nuestro propio estilo o utilizar el que viene por defecto en la librería.

  Los widgets son imágenes vectoriales. Una imagen vectorial se caracteriza por estar definida por atributos matemáticos. Un botón circular quedará definido por su centro y su radio. Mediante las imágenes vectoriales somos capaces de ampliar o reducir cualquier objetos sin perder resolución.

  Los widgets disponibles en las librerías son los siguientes: Button, Slider, Window, Check Box, Radio Button, Edit Box, List Box, Group Box, Horizontal/Vertical Scroll Bars, Progress Bar, Static Text, Picture, Dial, Meter.

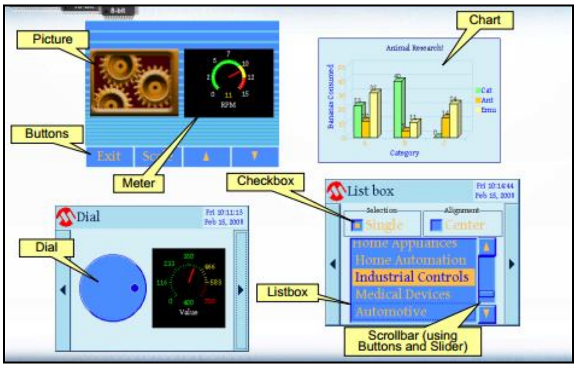


Figura 4.21. Ejemplos de widgets.

- Aplication Layer: Es la capa que interactuará con las capas inferiores. En el caso de nuestra aplicación se interactúa con el resto de capas, en concreto con los widgets, a través del control táctil.

4.6.4. Control de brillo.

4.6.4.1. Modulación por ancho de pulsos PWM.

La **modulación por ancho de pulsos (PWM)** es una técnica de modulación que controla el ciclo de trabajo de una señal periódica. De esta manera conseguimos aplicar una tensión media constante a lo largo del tiempo.

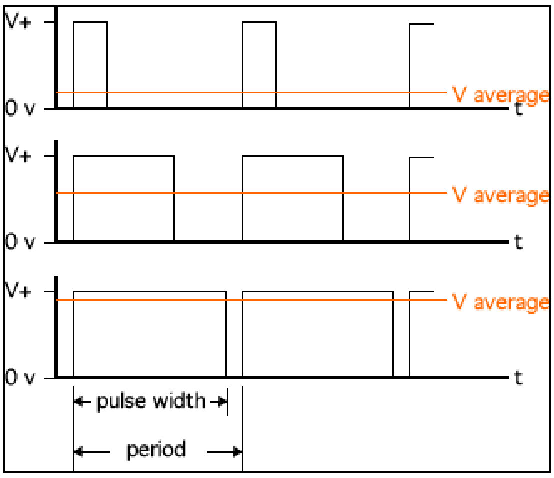
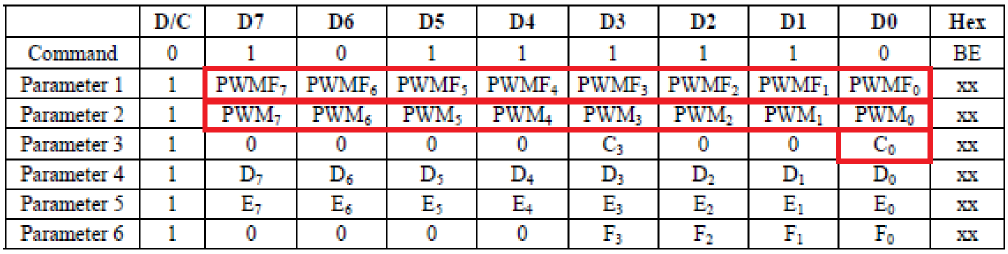


Figura 4.22. Modulación por ancho de pulsos.

Una de las características del controlador gráfico SSD1963 de nuestro sistema es la capacidad de controlar la retroiluminación a través de la modulación por ancho de pulsos (PWM). En este capítulo veremos la configuración software necesaria para el PWM.

La función del SSD1963 necesaria se conoce como set_pwm_conf y requiere de 6 parámetros de configuración que se detallan a continuación:



	D/C	D7	D6	D5	D4	D3	D2	D1	D0	Hex
Command	0	1	0	1	1	1	1	1	0	BE
Parameter 1	1	$PWMF_7$	$PWMF_6$	$PWMF_5$	$PWMF_4$	$PWMF_3$	$PWMF_2$	$PWMF_1$	$PWMF_0$	xx
Parameter 2	1	PWM_7	PWM_6	PWM_5	PWM_4	PWM_3	PWM_2	PWM_1	PWM_0	xx
Parameter 3	1	0	0	0	0	C_3	0	0	C_0	xx
Parameter 4	1	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	xx
Parameter 5	1	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	xx
Parameter 6	1	0	0	0	0	F_3	F_2	F_1	F_0	xx

Figura 4.23. Comando y parámetros de la función set_pwm_conf.

- PWMF[7:0]: configura la frecuencia de PWM.

$$PWMf = PLLf/(256 * (PWMF[7:0] + 1))/256$$

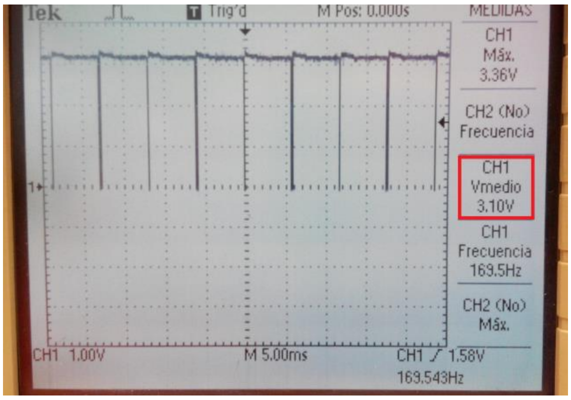
El controlador de brillo nos recomienda una frecuencia de PWM entre 100 y 500Hz. Con un PWMF[7:0]=8 y siendo PLLf=100MHz conseguiremos una frecuencia en torno a 170Hz.

- PWM[7:0]: Será el valor de la intensidad que queramos aplicar, siendo 0x00 el mínimo y 0xFF(255) el brillo máximo.

- De los parámetros restantes sólo nos interesa la habilitación del PWM con C[0] = 1. El resto de parámetros se dejarán en sus valores por defecto.

4.6.4.2. Comprobación de señales PWM en el osciloscopio.

La función que establece estos parámetros en el código se conoce como **void SetBacklight(intensity)**. A continuación se muestran capturas tomadas en el laboratorio del departamento de la señal con diferentes ciclos de trabajo del PWM, y por tanto con diferentes intensidades finales en la retroiluminación.

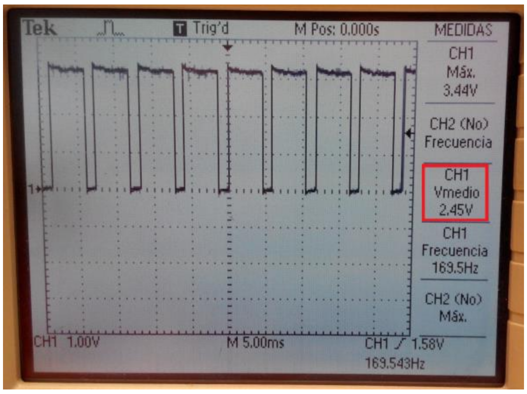


SetBacklight (250);

Ciclo de trabajo: 98% ;

Vmedio: 3,10V

Figura 4.24. Señal PWM. Ciclo de trabajo del 98%.

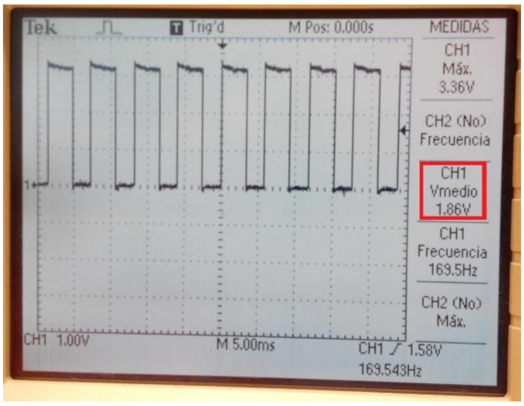


SetBacklight (200);

Ciclo de trabajo: 78%

Vmedio: 2,45V

Figura 4.25. Señal PWM. Ciclo de trabajo del 78%.

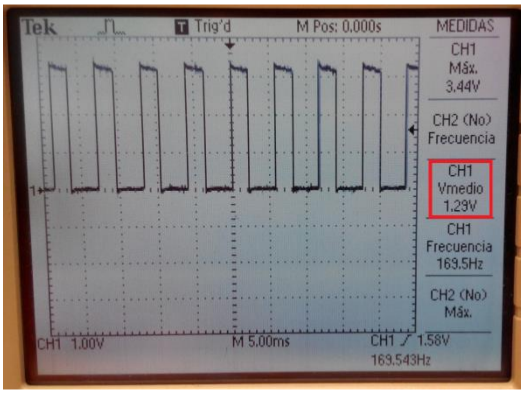


SetBacklight (150);

Ciclo de trabajo: 58.5%

Vmedio: 1,86V

Figura 4.26. Señal PWM. Ciclo de trabajo del 58.5%.

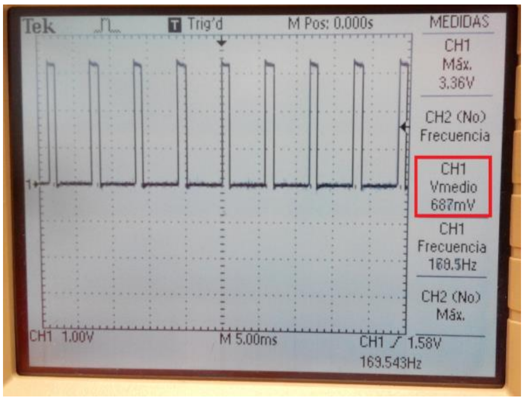


SetBacklight (100);

Ciclo de trabajo: 39%

Vmedio: 1,29V

Figura 4.27. Señal PWM. Ciclo de trabajo del 39%.



SetBacklight (50);

Ciclo de trabajo: 19.5%

Vmedio: 687mV

Figura 4.28. Señal PWM. Ciclo de trabajo del 19.5%.

4.6.5. Diagrama de estados del programa principal.

La utilización de las librerías gráficas utiliza un código formado por varias funciones gráficas entre las que se encadenan las entradas de usuario. La estructura general de una secuencia de iniciación gráfica es la siguiente:

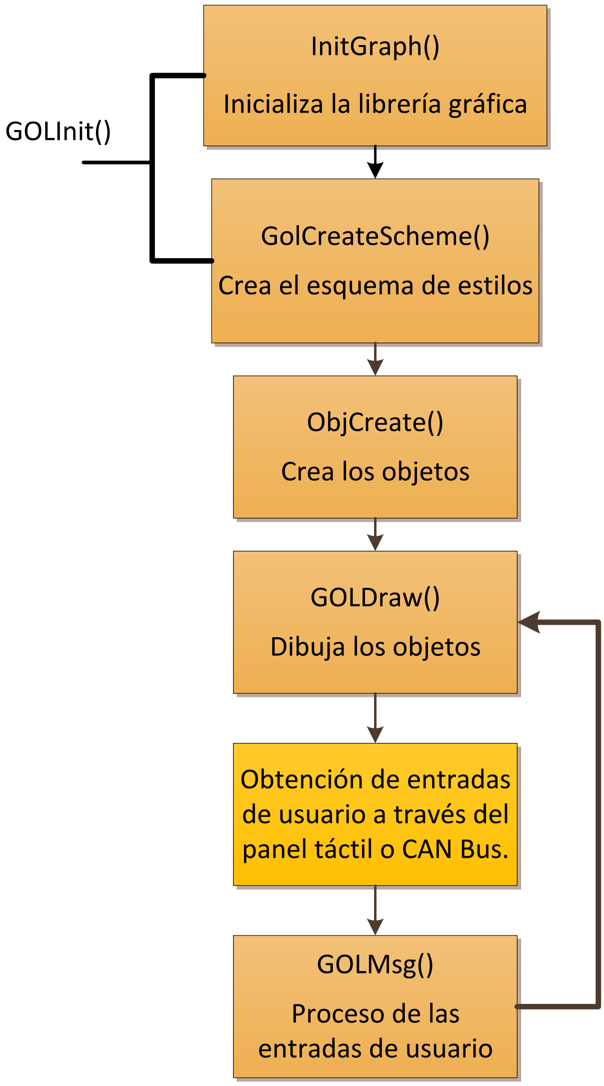


Figura 4.29. Diagrama de estados librería gráfica.

1. Inicialización de la librería gráfica.

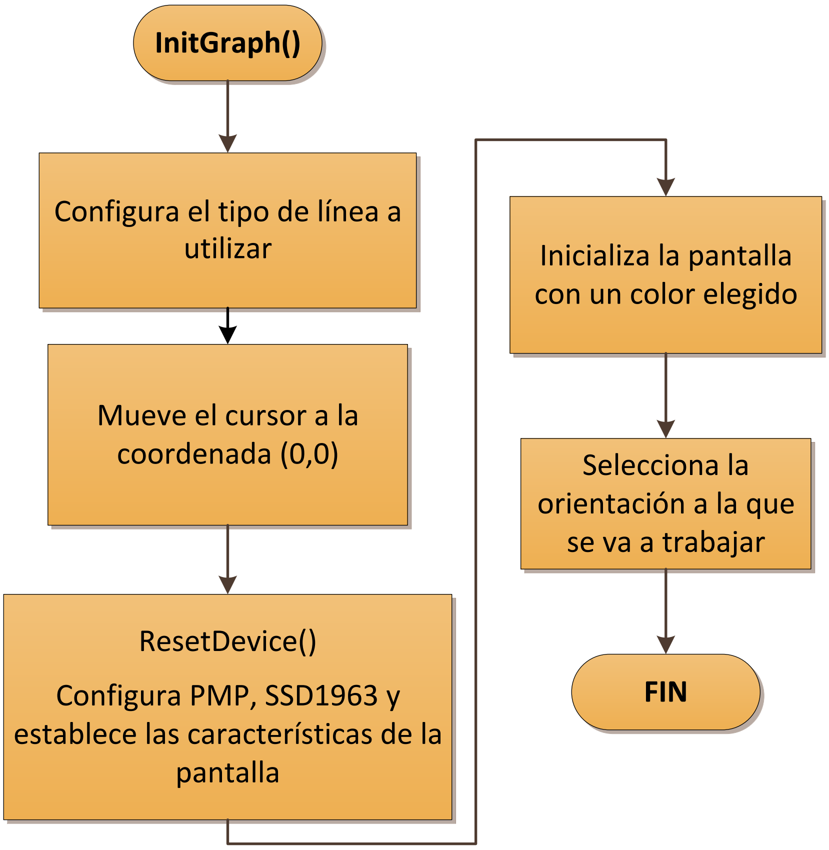


Figura 4.30. Diagrama de estados. InitGraph.

2. Creación de esquema de estilos. En este punto se crean las diferentes **paletas de colores** que se utilizarán en la aplicación. En caso de no crear ningún estilo se utilizará el esquema por defecto. Los esquemas de estilos incluyen los colores que se utilizarán posteriormente en los widgets, como color principal, color secundario, color de widget deshabilitado, color de fondo, color de biselado, etc. Además incluye el **tipo de letra** que se utilizará.

3. Antes de dibujar los objetos en la pantalla se crea una lista enlazada con los objetos a dibujar en cada una de las ventanas. Previamente se libera la memoria de la lista anteriormente creada mediante GOLFree(). Esta lista contendrá cada uno de los objetos y sus características en cuanto a forma, posición, texto y esquema de estilos propio de cada uno.

4. En este momento se dibujan uno a uno los objetos almacenados en memoria. Por tanto, se dibuja la ventana en la que nos encontremos en ese momento. En este momento la aplicación pasa de un estado de creación (CREAR_MENU_PRINCIPAL, CREAR_MENU_OPCIONES o CREAR_MENU_TEST) a un estado de gestión de la ventana. En siguientes iteraciones su labor es la de chequear los parámetros de los objetos de la lista enlazada para comprobar si tienen que ser redibujados o no.

5. En este momento nos encontramos con los estados de gestión del programa (DISPLAY_MENU_PRINCIPAL, DISPLAY_MENU_OPCIONES o DISPLAY_MENU_TEST). En estos estados el programa se queda a la espera a que se produzca algún evento. Estos eventos pueden ser mediante una pulsación o mediante la recepción de algún dato del CAN Bus.

6. Se procesan y gestionan las entradas de usuario. Por ejemplo, si nos encontramos en el Menú Principal y pulsamos en Opciones, pasaremos a un nuevo estado de creación (punto 4). O si recibimos un dato del CAN Bus de algún sensor, actualizaremos el parámetro del widget correspondiente, que será redibujado al volver al punto 4.

4.6.6. Ventanas creadas

Dado que no se ha trabajado con el sistema en el propio vehículo, se han creado 3 ventanas para mostrar información y comprobar el funcionamiento de los componentes que conforman el proyecto. En futuros proyectos no es necesario la modificación de ningún componente hardware. Por tanto, mediante software se podrá configurar la aplicación según el ingeniero/piloto desee. Las 3 ventanas que se han creado cuentan con el mismo estilo gráfico pero diferente información. En la parte superior izquierda de cada una de ellas se informa mediante texto de la ventana en la que se encuentra el usuario. En la parte inferior se han creado dos botones para ir a cualquiera de las otras dos ventanas restantes.

4.6.6.1. Menú Principal.

Esta ventana se encarga de dar al piloto toda la información que pueda ser relevante durante las diferentes pruebas de la Fórmula SAE. Para la información se han utilizado diferentes widgets y esquemas de estilos. La información que se muestra al piloto es la siguiente:

- Información del menú en el que se encuentra con el título en la parte superior izquierda.
- Velocidad. Mediante un velocímetro creado a partir del widget Meter. Se muestra la velocidad en km/h tanto en número como a través de la aguja del velocímetro.
- Revoluciones por minuto (RPM). De las misma manera que con la velocidad se muestran las revoluciones en número (x1000) y mediante la aguja del widget.
- Medidores de temperatura de agua y de aceite. En % de la temperatura máxima admitida por el sistema.
- Medidor de la carga de la batería en %.

- Botones para acceder al menú de test y al menú de opciones en la parte inferior de la pantalla.

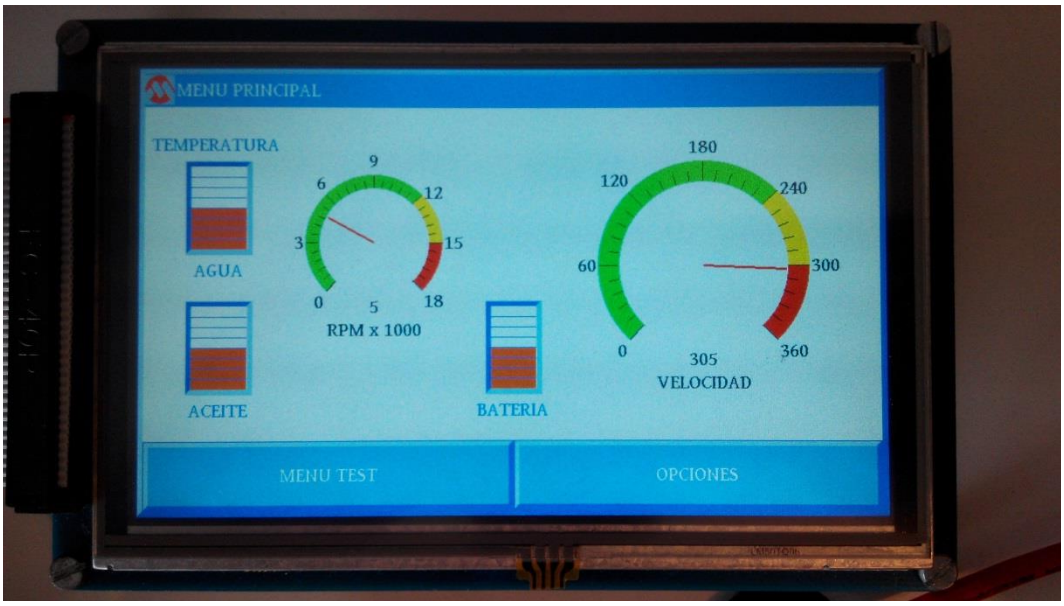


Figura 4.31. Ventana Menú Principal.

4.6.6.2. Menú Opciones

El menú de opciones servirá tanto para configurar opciones básicas como el idioma o el brillo como para seleccionar opciones en la conducción. La información que se muestra es la siguiente:

- Información del menú en el que se encuentra con el título en la parte superior izquierda.
- Configuración del brillo. A través de este widget actuamos directamente en el PWM del controlador gráfico, y por tanto en la retroiluminación del sistema.
- Configuración de idioma. Se han establecido dos idiomas para la aplicación: español e inglés.
- 3 botones que enviarán una trama CAN para la petición de alguna opción referente a la conducción. En nuestro caso

sólo enviamos la trama de petición, será otro sistema el que actúe una vez reciba e interprete nuestro mensaje.

- Botones para acceder al menú principal y al menú de test en la parte inferior de la pantalla.

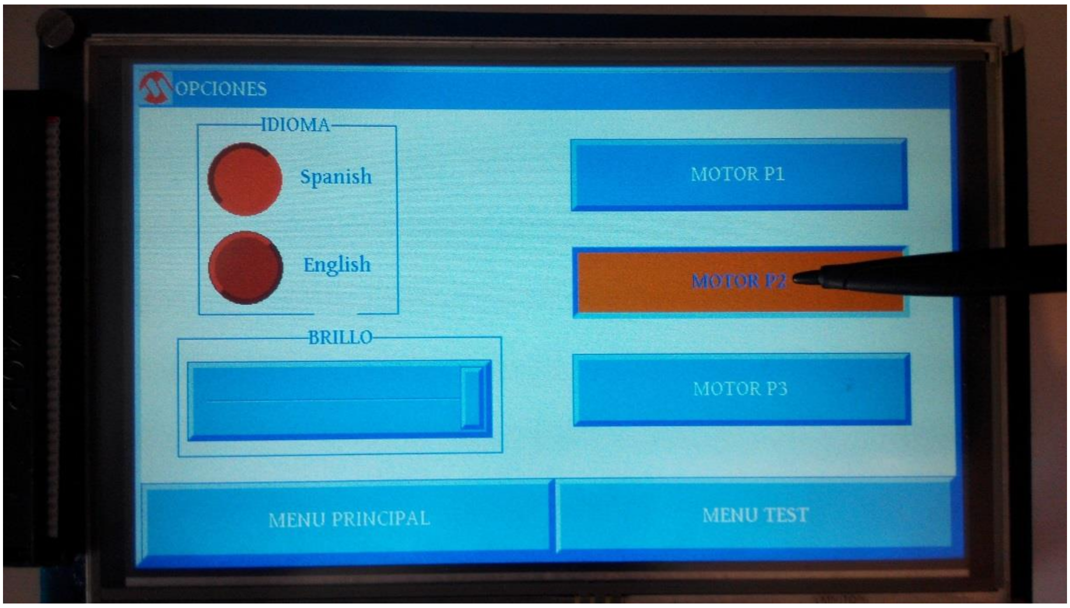


Figura 4.32. Ventana Menú Opciones.

4.6.6.3. Menú Test.

Este menú está especialmente indicado para utilizarse durante la elaboración de pruebas previas a la competición. Se podrá comprobar la recepción de tramas CAN del sistema, la representación gráfica de alguna variable que pueda resultar útil a los ingenieros, incluso la comprobación del funcionamiento táctil. Cuenta con las siguientes opciones:

- Información del menú en el que se encuentra con el título en la parte superior izquierda.
- Se visualizarán las tramas CAN con el identificador correspondiente aceptado por nuestro sistema. Las visualización se divide en 5 datos que corresponden a

RPM, temperatura del agua, temperatura del aceite, batería y velocidad.

- Widget para la comprobación del correcto funcionamiento del panel táctil.
- Visualización de una gráfica creada por defecto. Esta gráfica podría utilizarse, una vez conectados otros sistemas al CAN Bus, como visualización directa de cualquier variable mediante las pruebas del coche, ya sea posición del acelerador, velocidad, consumo, etc.
- Botones para acceder al menú principal y al menú de opciones en la parte inferior de la pantalla.

RPM x 1000	10.4	D0
TEMPERATURA	48	D1
ACEITE	15	D2
BATERIA	89	D3
VELOCIDAD	245	D4D5

Figura 4.33. Ventana Menú Test.

5. CONSUMO DEL SISTEMA COMPLETO.

Para la medición del consumo del sistema completo se han realizado medidas con el programa en ejecución y todos los sistemas conectados, tanto el simulador de CAN Bus como el programador PICKit3.

En la siguiente imagen podemos observar el mayor consumo del sistema en condiciones de iluminación máxima.

Figura 5.1. Consumo del sistema completo.

La potencia resultante en este caso es de

$$[P = V \times I = 13{,}5V \times 240mA = 3.24W]$$

Posteriormente se observó que la retroiluminación LED es uno de los elementos que más influyen en este aspecto. Por tanto, se tomaron medidas del sistema en funcionamiento evaluando todo el rango de iluminación.

Como resultado se ha observado una diferencia entre la retroiluminación máxima y mínima de **1.188W**, que corresponden en porcentaje al **36.6%** de la potencia total.

V=13,5V

Iluminación %	Backlight(X)	I(mA)	P(W)=V·I
100	255	240	3,24
98	250	238	3,213
94	240	234	3,159
90	230	229	3,0915
86	220	226	3,051
82	210	222	2,997
78	200	218	2,943
75	190	214	2,889
71	180	208	2,808
67	170	203	2,7405
63	160	200	2,7
59	150	196	2,646
55	140	192	2,592
51	130	188	2,538
47	120	182	2,457
43	110	178	2,403
39	100	175	2,3625
35	90	171	2,3085
31	80	166	2,241
27	70	162	2,187
24	60	159	2,1465
20	50	157	2,1195
16	40	156	2,106
12	30	154	2,079
8	20	153	2,0655
4	10	152	2,052

Figura 5.2. Tabla de consumo.

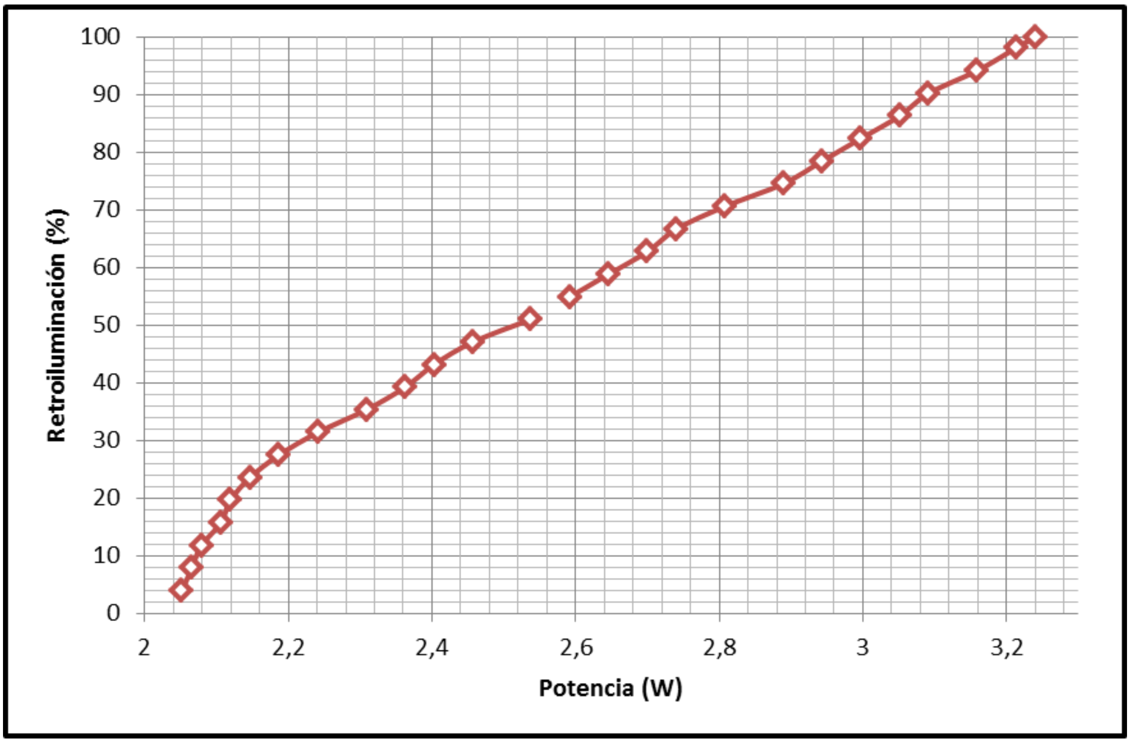


Figura 5.3. Evolución del consumo.

6. PRESUPUESTO.

REFERENCIA	DESCRIPCIÓN	COSTE UNITARIO	TOTAL
* 1778492	PIC32MX795F512L	9,11	9,11
* 1578407	MCP1825S-5V	0,448	0,448
* 1578404	MCP1825S-3,3V	0,588	0,588
* 9758569	MCP2551 CAN Transceiver	0,788	0,788
**	PCB Pantalla + Panel Táctil	46,63	46,63
1755268	TL2575 ADJ	2,2	2,2
4218164	3x Jumpers	0,199	0,597
1842346RL	XTAL 8MHz	0,587	0,587
1929752	Inductor	2,31	2,31
8391246	Conector D-SUB 9	1,91	1,91
1099242	2x Abrazadera 40 vías	0,82	1,64
8395977	2x Conector 40 vías	2,11	4,22
1642041	RJ11	0,73	0,73
1841228	10 x vías 2,54mm	1,18	1,18
1792766	Header 5,08mm 2Way	0,66	0,66
1625259	Diodo Schottky	0,54	0,54
297379	Cable plano	3,85	3,85
3204923	Sustrato PCB Doble Cara	10,51	10,51
1466832	8 x Separador	0,26	2,08
1692596	Potenciómetro 5Kohm	2,59	2,59
1650896	2x Condensador 20pF	0,217	0,434
9451838	Condensador Elect. 100uF	0,138	0,138
9451730	Condensador Elect. 220uF	0,183	0,183
1759434	1x Condensador 10uF	0,059	0,059
1759444	2x Condensador 4,7uF	0,045	0,09
9227873	2x Condensador 1uF	0,084	0,168
1759297	7 x Condensadores 0,1uF	0,036	0,252
1469970	Resistencia 10kohm	0,025	0,025
		SUBTOTAL 1	94,52

Figura 6.1. Presupuesto de componentes.

*Estos componentes fueron adquiridos gratuitamente mediante la petición de muestras gratuitas a Microchip Direct.

** El módulo Pantalla + Panel Táctil + SSD1963 fue adquirido en la siguiente referencia de un vendedor de productos electrónicos en el portal Ebay.

http://www.ebay.es/itm/5-TFT-LCD-Module-Display-Touch-Panel-Screen-PCB-Adapter-Build-in-SSD1963-/221044250453?pt=LH_DefaultDomain_0&hash=item3377439755#ht_5744wt_906

El precio del producto en el momento en que se adquirió fue de 46.63€.

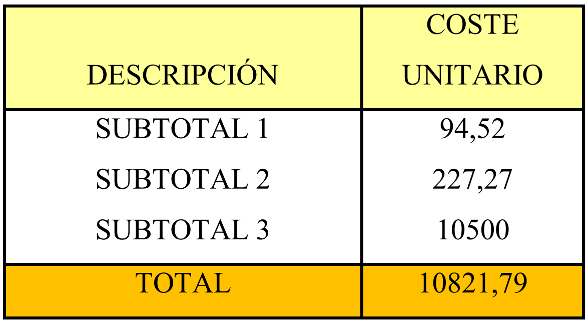
El **coste total del prototipo** desarrollado es de 94.52 tomándose solamente en cuenta el precio de los componentes.

No se incluyen por tanto en el coste del prototipo todo el material necesario para su desarrollo que fue proporcionado por el departamento. En el siguiente subtotal se incluyen los programadores del microcontrolador y el simulador de CAN Bus. Por último se hace un cálculo aproximado de las horas invertidas en el desarrollo del proyecto tomando el coste de hora de ingeniería a 10€.

DESCRIPCIÓN	COSTE UNITARIO	TOTAL
ICD3	150,02	150,02
PICKIT 3	35,98	35,98
CAN Bus Monitor Demo	41,27	41,27
	SUBTOTAL 2	227,27

DESCRIPCIÓN	Horas	TOTAL
Estudio inicial	150	1500
Desarrollo Hardware	250	2500
Desarrollo Software	350	3500
Depuración y validación	300	3000
	SUBTOTAL 3	10500

Figura 6.2. Presupuestos de kits y horas de ingeniería.



DESCRIPCIÓN	COSTE UNITARIO
SUBTOTAL 1	94,52
SUBTOTAL 2	227,27
SUBTOTAL 3	10500
TOTAL	10821,79

Figura 6.3. Presupuestos total.

7. CONCLUSIONES

En cuanto a las conclusiones que puedo extraer una vez finalizado el proyecto se podrían dividir en varios aspectos.

En el aspecto basado en los objetivos propuestos puedo concluir que se han cumplido todos los objetivos marcados. Los objetivos se han cumplido en mayor o menor medida, contando por ejemplo con inconvenientes ajenos al propio alumno y tutor, como son la falta de presupuesto para realizar una tarjeta profesional completa. Tampoco se han podido realizar pruebas en el vehículo en cuestión dado que no era algo que dependiera exclusivamente del departamento. Aun así, considero cumplidos los objetivos marcados.

En el aspecto personal destacaría todas las capacidades obtenidas durante el desarrollo del mismo. La elección de este proyecto y como considero personalmente que deberían ser los proyectos de final de carrera, es ser capaz de aplicar la mayor parte de conceptos aprendidos durante estos años en un sistema de cierta complejidad, aunque esto repercuta en que el proyecto tenga mayor dificultad o duración. Además de aplicar conceptos, también ser capaz de demostrar las capacidades de ingeniero al enfrentarse a situaciones o conceptos no conocidos por el alumno hasta el momento.

Considero por tanto que he sido capaz de aplicar conceptos aprendidos previamente, por ejemplo en el diseño y elaboración de PCBs o el manejo de microprocesadores por poner unos ejemplos. Además he sido capaz de enfrentarme a todo tipo de “desafíos” tecnológicos como podrían ser el conocer el funcionamiento de un panel táctil, aprender desde cero a utilizar una familia nueva de microcontroladores, el procesado de imágenes a través de un controlador gráfico o el protocolo de comunicación CAN Bus por citar algunos.

De todo esto y haciendo balance de todas experiencias vividas durante el transcurso del proyecto, tanto positivas que te animan a seguir adelante como negativas que son necesarias de atravesar tanto en un trabajo como éste como en la propia vida, considero el desarrollo de este proyecto una experiencia satisfactoria a nivel profesional y personal.

8. LINEAS FUTURAS DE TRABAJO.

Considero que este proyecto tiene bastantes capacidades para desarrollarse futuros proyectos partiendo del mismo. Las recomendaciones en cuanto a líneas futuras serían las siguientes.

- La creación de una PCB de forma profesional y la incorporación al volante del vehículo de Fórmula SAE para la posibilidad de realizar pruebas de la pantalla en condiciones de luminosidad, temperatura, humedad, etc.
- El desarrollo hardware y software necesario para la utilización de la tarjeta SD ya incluida en la segunda PCB. Con la utilización de la tarjeta SD se podrían incluir imágenes del circuito de pruebas. Además se podría realizar la grabación de datos y de gráficas durante una sesión de entrenamiento para su posterior estudio.
- Conexión de la pantalla con sensores reales del vehículo y otros subsistemas para realizar una gestión de errores del CAN Bus o comprobar las selecciones en la conducción a petición del piloto.

9. BIBLIOGRAFÍA

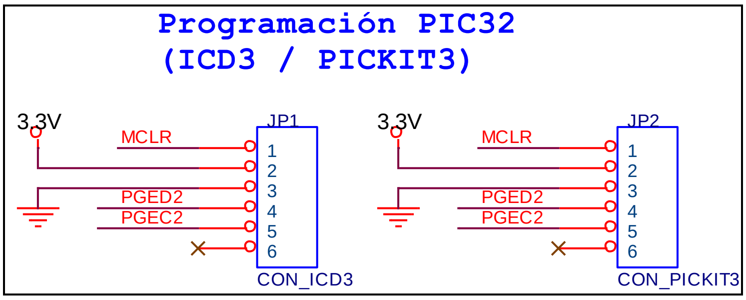
Además de todos los datasheets específicos de cada componente, se han utilizado numerosos artículos disponibles en internet, así como revistas científicas y proyectos de temas concretos como el CAN Bus o las pantallas táctiles. Algunos ejemplos se presentan a continuación.

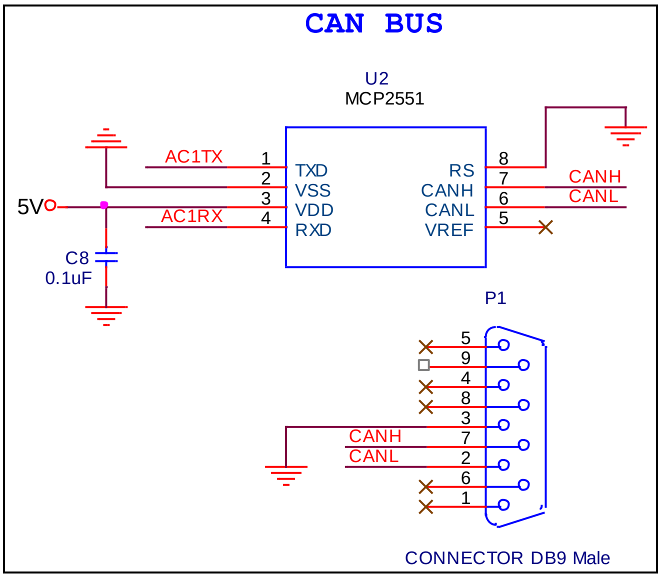
- http://ingeniatic.euitt.upm.es/index.php/tecnologias/item/542-pantalla-t%C3%A1ctil
- http://www.netambulo.com/2012/11/13/tipos-de-pantallas-tactiles-resistivas-capacitivas-e-infrarrojas/
- http://www.convertronic.net/Pantallas-Tactiles/pantallas-tactiles.html
- http://www.mikroe.com/downloads/get/488/es_mikroe_article_pascal_pic_01_09.pdf
- http://www.tecnogeek.com/verpost.php?id_noticia=817
- http://www.ecojoven.com/dos/05/tactil.html
- http://www.eizo.com/global/library/basics/basic_understanding_of_touch_panel/
- http://www.elotouch.com/Technologies/compare_resist.asp
- http://www.fayerwayer.com/2011/11/el-origen-de-la-pantalla-tactil/
- http://microcontroladoressesv.wordpress.com/arquitectura-de-los-microcontroladores/
- http://www.ecured.cu/index.php/Microcontroladores_PIC.
- http://es.wikipedia.org/wiki/Microcontrolador_PIC#Caracter.C3.ADsticas
- http://www.mikroe.com/chapters/view/79/capitulo-1-el-mundo-de-los-microcontroladores/
- http://www.elemon.com.ar/media/catalogos/z%20Boletines%20tecnicos/Introducci%C3%B3n%20a%20la%20programaci%C3%B3n%20de%20PIC32%20con%20ARDUINO.pdf
- http://www.aquihayapuntes.com/curso-pic32.html
- http://www.microchip.com/stellent/groups/SiteComm_sg/documents/DeviceDoc/en542879.pdf

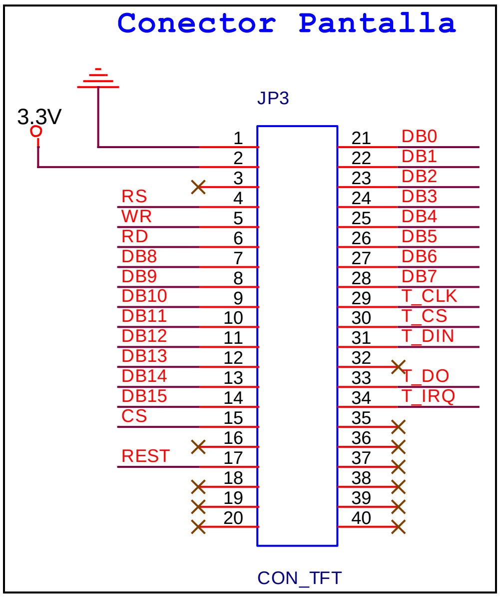
- http://tecbolivia.com/index.php/articulos-y-tutoriales-microcontroladores/19-icsp-como-usar-qprogramacion-serial-en-circuitoq-con-microcontroladores-pic
- http://picfernalia.blogspot.com.es/2013/04/comunicaciones-serie-spi.html
- http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/
- https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all
- http://www.microchip.com/Developmenttools/ProductDetails.aspx?PartNO=MCP2515DM-BM#dtDocumentation

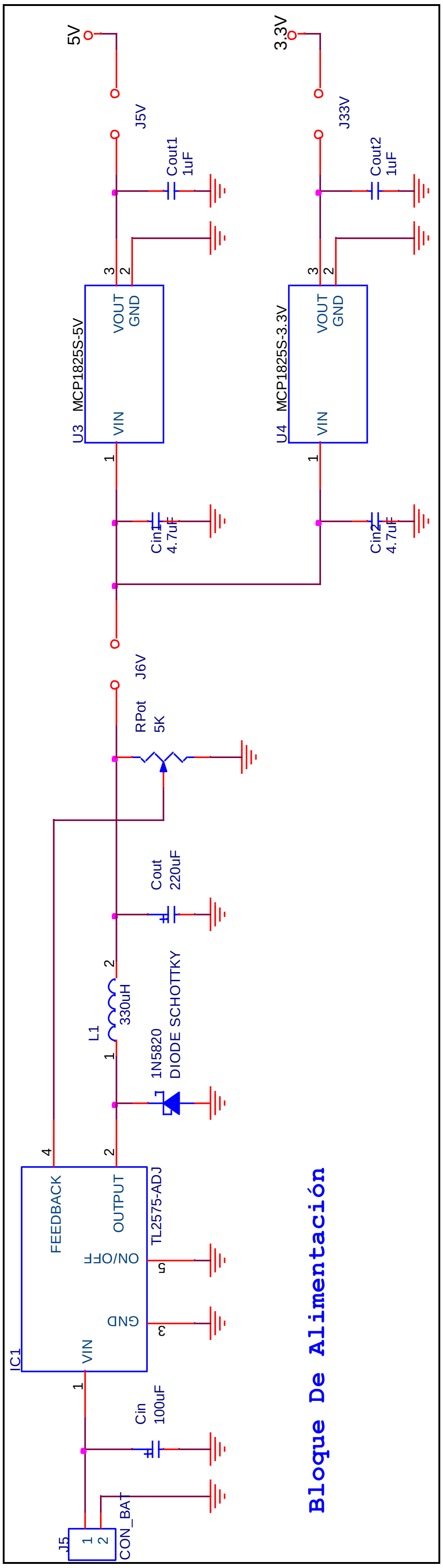
ANEXO 1

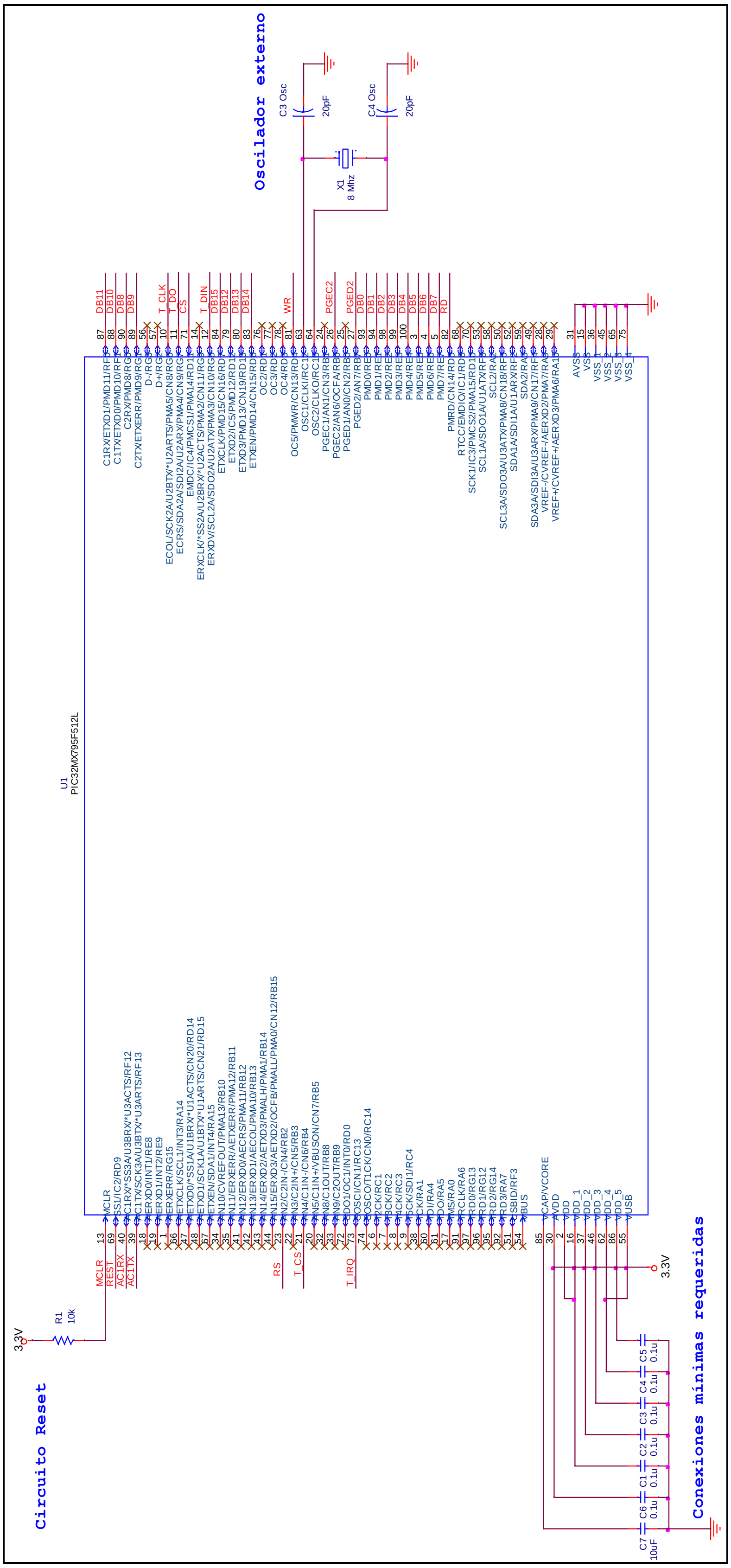
ESQUEMÁTICOS

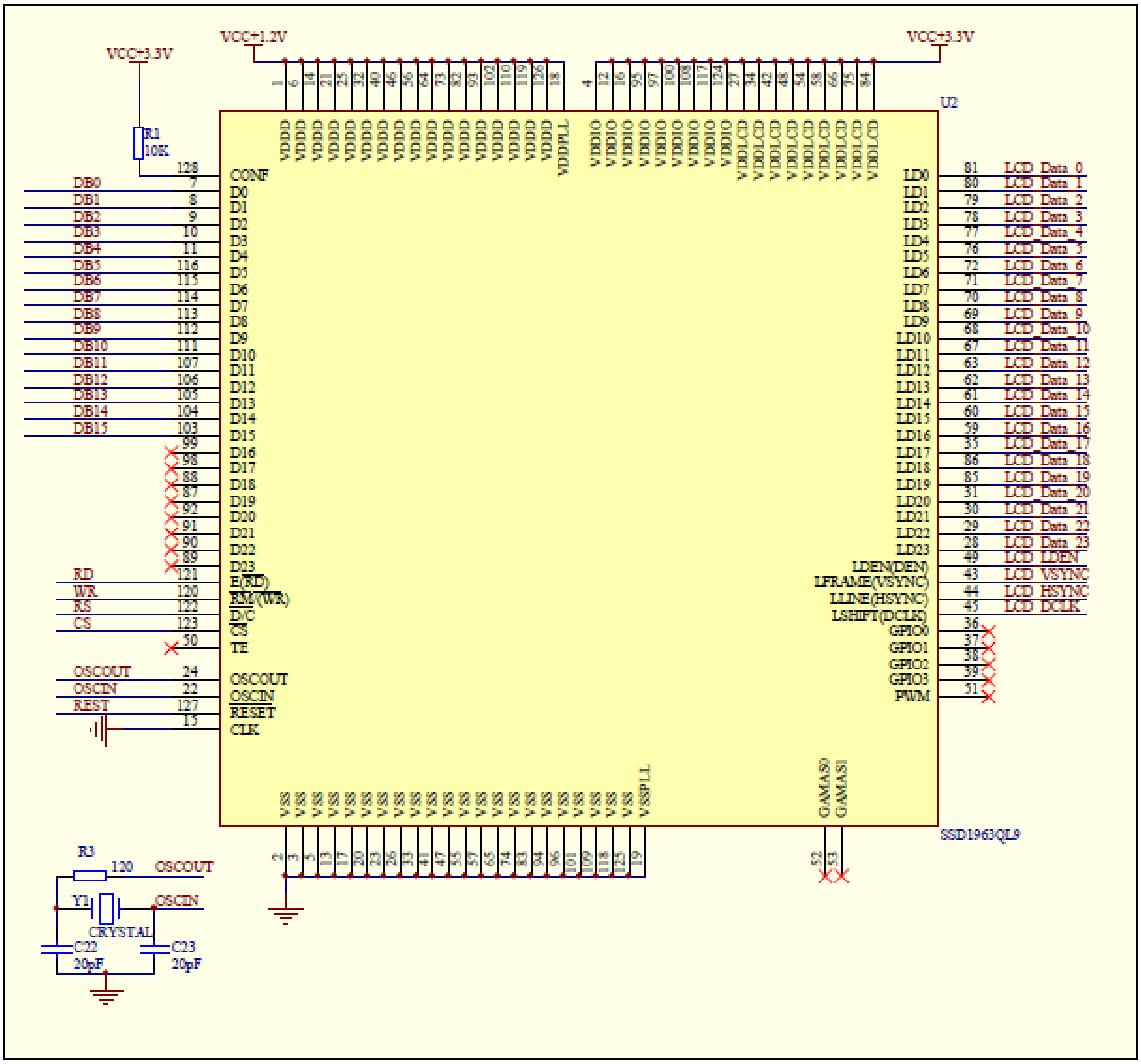

Programación PIC32
(ICD3 / PICKIT3)
3.3V
MCLR
PGED2
PGEC2
JP1
1
2
3
4
5
6
CON_ICD3
3.3V
MCLR
PGED2
PGEC2
JP2
1
2
3
4
5
6
CON_PICKIT3

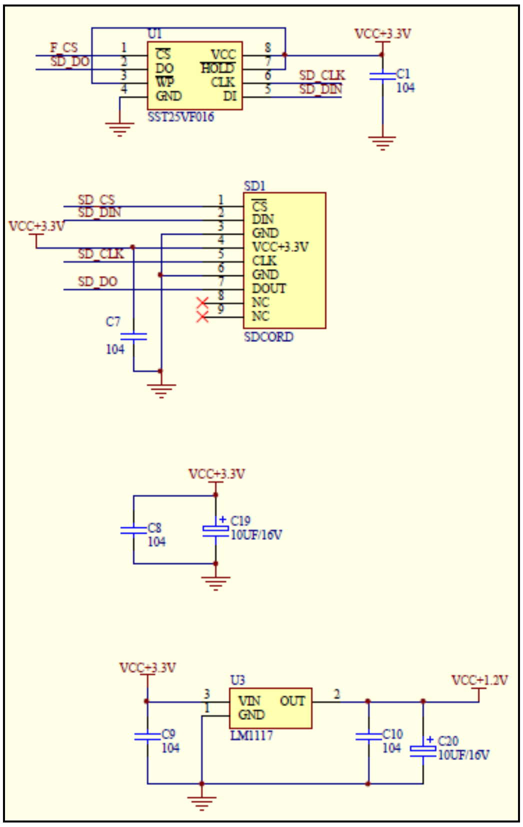

CAN BUS
U2
MCP2551
AC1TX
AC1RX
5V
C8
0.1uF
1 TXD
2 VSS
3 VDD
4 RXD
8 RS
7 CANH
6 CANL
5 VREF
CANH
CANL
P1
5
9
4
8
3
7
2
6
1
CANH
CANL
CONNECTOR DB9 Male

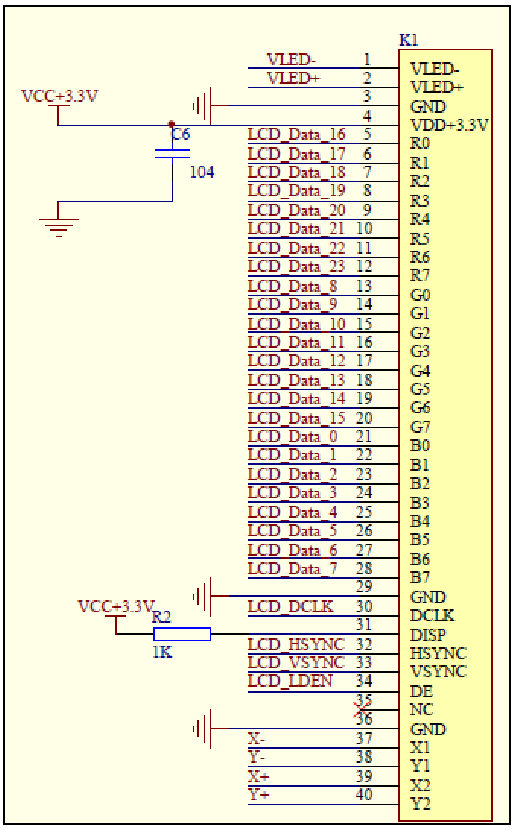

Conector Pantalla
3.3V
JP3
1
2
3
RS 4
WR 5
RD 6
DB8 7
DB9 8
DB10 9
DB11 10
DB12 11
DB13 12
DB14 13
DB15 14
CS 15
16
REST 17
18
19
20
21 DB0
22 DB1
23 DB2
24 DB3
25 DB4
26 DB5
27 DB6
28 DB7
29 T_CLK
30 T_CS
31 T_DIN
32
33 T_DO
34 T_IRQ
35
36
37
38
39
40
CON_TFT

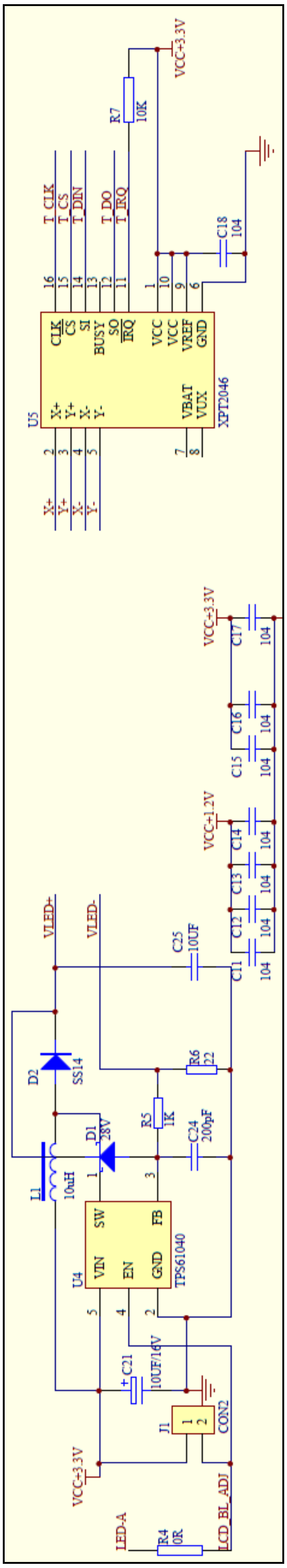

Bloque De Alimentación
5V
3.3V
J5V
J33V
Cout1
1uF
Cout2
1uF
U3
MCP1825S-5V
U4
MCP1825S-3.3V
VOUT
GND
VIN
Cin1
4.7uF
Cin2
4.7uF
J6V
RPot
5K
Cout
220uF
L1
330uH
1N5820
DIODE SCHOTTKY
IC1
TL2575-ADJ
FEEDBACK
OUTPUT
ON/OFF
GND
VIN
Cin
100uF
J5
CON_BAT
1
2
3
4
5

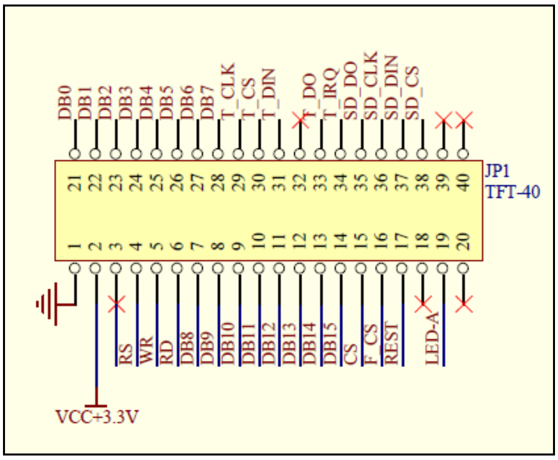

Oscilador externo
C3 Osc
20pF
C4 Osc
20pF
X1
8 Mhz
U1
PIC32MX795F512L
Circuito Reset
3.3V
R1
10k
MCLR
REST
AC1RX
AC1TX
RS
T_CS
T_IRQ
DB11
DB10
DB8
DB9
T_CLK
T_DO
CS
T_DIN
DB15
DB12
DB13
DB14
WR
PGEC2
PGED2
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
RD
AVSS
VSS
VDD
VUSB
VCAP/VCORE
AVDD
VBUS
Conexiones mínimas requeridas
C1
C2
C3
C4
C5
C6
0.1u
C7
10uF


VCC+3.3V
VCC+1.2V
VCC+3.3V
R1
10K
U2
SSD1963QL9
VDDD
VDDPLL
VDDIO
VDDLCD
1 6 14 21 25 32 40 46 56 64 73 82 93 102 110 119 126 18
4 12 16 95 97 100 108 117 124 27 34 42 48 54 58 66 75 84
128 CONF
DB0 7 D0
DB1 8 D1
DB2 9 D2
DB3 10 D3
DB4 11 D4
DB5 116 D5
DB6 115 D6
DB7 114 D7
DB8 113 D8
DB9 112 D9
DB10 111 D10
DB11 107 D11
DB12 106 D12
DB13 105 D13
DB14 104 D14
DB15 103 D15
99 D16
98 D17
88 D18
87 D19
92 D20
91 D21
90 D22
89 D23
RD 121 E(RD)
WR 120 RW(WR)
RS 122 D/C
CS 123 CS
50 TE
OSCOUT 24 OSCOUT
OSCIN 22 OSCIN
REST 127 RESET
15 CLK
LD0 81 LCD_Data_0
LD1 80 LCD_Data_1
LD2 79 LCD_Data_2
LD3 78 LCD_Data_3
LD4 77 LCD_Data_4
LD5 76 LCD_Data_5
LD6 72 LCD_Data_6
LD7 71 LCD_Data_7
LD8 70 LCD_Data_8
LD9 69 LCD_Data_9
LD10 68 LCD_Data_10
LD11 67 LCD_Data_11
LD12 63 LCD_Data_12
LD13 62 LCD_Data_13
LD14 61 LCD_Data_14
LD15 60 LCD_Data_15
LD16 59 LCD_Data_16
LD17 55 LCD_Data_17
LD18 86 LCD_Data_18
LD19 85 LCD_Data_19
LD20 31 LCD_Data_20
LD21 30 LCD_Data_21
LD22 29 LCD_Data_22
LD23 28 LCD_Data_23
LDEN(DEN) 49 LCD_LDEN
LFRAME(VSYNC) 43 LCD_VSYNC
LLINE(HSYNC) 44 LCD_HSYNC
LSHIFT(DCLK) 45 LCD_DCLK
GPIO0 36
GPIO1 37
GPIO2 38
GPIO3 39
PWM 51
VSS
VSSPLL
2 3 5 13 17 20 23 26 33 41 47 55 57 65 74 83 94 96 101 109 118 125 19
GAMAS0 52
GAMAS1 53
R3
120
OSCOUT
Y1
OSCIN
CRYSTAL
C22
20pF
C23
20pF


U1
SST25VF016
F_CS
SD_DO
CS
DO
WP
GND
VCC
HOLD
CLK
DI
1
2
3
4
8
7
6
5
SD_CLK
SD_DIN
VCC+3.3V
C1
104
SD1
SD_CS
SD_DIN
VCC+3.3V
SD_CLK
SD_DO
CS
DIN
GND
VCC+3.3V
CLK
GND
DOUT
NC
NC
9
SDCORD
C7
104
VCC+3.3V
C8
104
C19
10UF/16V
VCC+3.3V
U3
VIN
OUT
GND
LM1117
C9
104
C10
104
C20
10UF/16V
VCC+1.2V


K1
VLED-
VLED+
VCC+3.3V
C6
104
LCD_Data_16
LCD_Data_17
LCD_Data_18
LCD_Data_19
LCD_Data_20
LCD_Data_21
LCD_Data_22
LCD_Data_23
LCD_Data_8
LCD_Data_9
LCD_Data_10
LCD_Data_11
LCD_Data_12
LCD_Data_13
LCD_Data_14
LCD_Data_15
LCD_Data_0
LCD_Data_1
LCD_Data_2
LCD_Data_3
LCD_Data_4
LCD_Data_5
LCD_Data_6
LCD_Data_7
LCD_DCLK
VCC+3.3V
R2
1K
LCD_HSYNC
LCD_VSYNC
LCD_LDEN
X-
Y-
X+
Y+
VLED-
VLED+
GND
VDD+3.3V
R0
R1
R2
R3
R4
R5
R6
R7
G0
G1
G2
G3
G4
G5
G6
G7
B0
B1
B2
B3
B4
B5
B6
B7
GND
DCLK
DISP
HSYNC
VSYNC
DE
NC
GND
X1
Y1
X2
Y2


VCC+3.3V
R7
10K
T_CLK
T_CS
T_DIN
T_DO
T_IRQ
C18
104
U5
XPT2046
X+
Y+
X-
Y-
CLK
CS
SI
BUSY
SO
IRQ
VCC
VCC
VREF
GND
VBAT
VUX
VCC+3.3V
C17
104
C16
104
C15
104
VCC+1.2V
C14
104
C13
104
C12
104
C11
104
VLED+
VLED-
C25
10UF
D2
SS14
R6
22
R5
1K
D1
28V
C24
200pF
L1
10uH
U4
TPS61040
SW
VIN
EN
GND
FB
C21
10UF/16V
J1
CON2
VCC+3.3V
LED-A
R4
0R
LCD_BL_ADJ


JP1
TFT-40
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
T_CLK
T_CS
T_DIN
T_DO
T_IRQ
SD_DO
SD_CLK
SD_DIN
SD_CS
RS
WR
RD
DB8
DB9
DB10
DB11
DB12
DB13
DB14
DB15
CS
F_CS
REST
LED-A
VCC+3.3V

ANEXO 2

MANUAL DE USUARIO

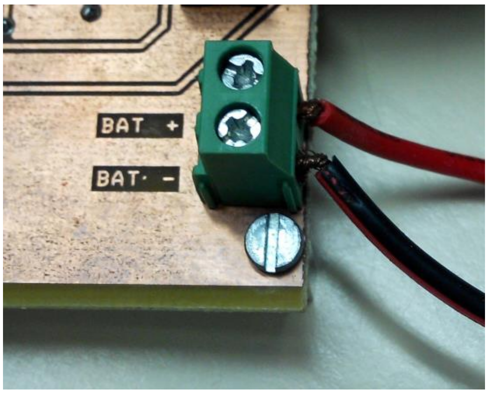
1. CONTENIDO DEL SISTEMA

El sistema final está compuesto por dos placas de circuito impreso unidas por un cable plano.

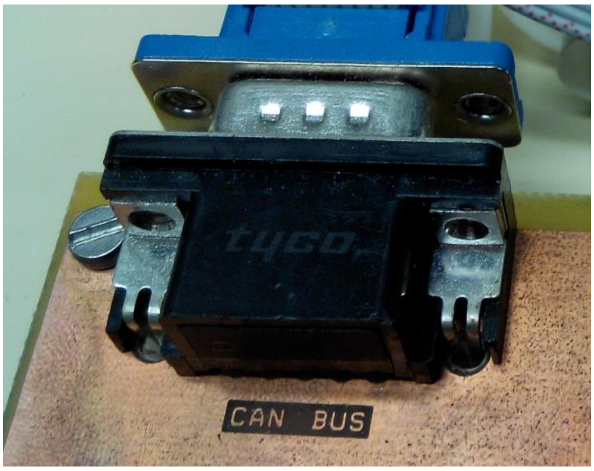
1.1. PCB_1: Es la encargada de la gestión de la aplicación. Está formada por el microcontrolador PIC con sus conexiones básicas y los bloques de programación, de alimentación y de CAN Bus,.

Los conectores disponibles son:

- **Conector de alimentación:** El conector de alimentación con las tomas positiva y negativa con conexión mediante tornillos pasantes. El circuito de alimentación está diseñado para una entrada de entre 12V y 13.5V.

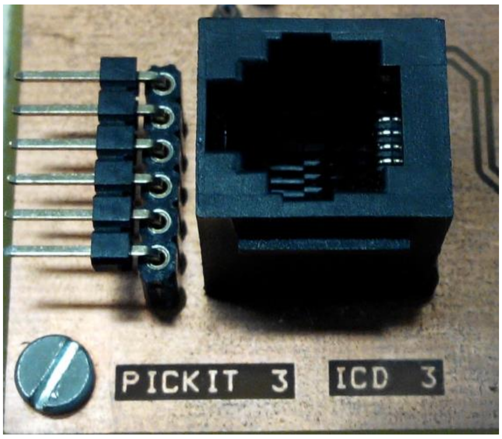


- **Conector de CAN Bus:** Se trata de un conector macho DB-9. Se utiliza para conectar una de las tarjetas del simulador CAN Bus.

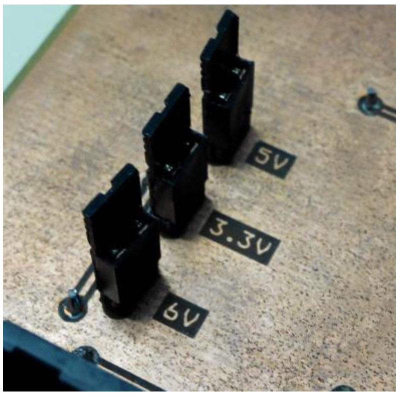


- **Conectores de programación:**

  Conectores diseñados para la conexión de los programadores / depuradores del PIC. No se podrán conectar los dos programadores al mismo tiempo. El conector izquierdo de la imagen formado por 6 pines en codo se conecta al PICKit 3 mientras que el conector RJ-11 se conecta al ICD 3.



- **Puntos de test (Jumpers):** La PCB 1 cuenta con 3 jumpers en el circuito de alimentación en los puntos de voltaje de 6V, 5V y 3.3V tanto como protección del resto del sistema como para servir de puntos de test.



- **Conector a PCB_2.** Se trata de un conector de 40 pines utilizado para conectarse al módulo formado por la pantalla y el controlador táctil.

1.2. PCB_2: Contiene la pantalla y el panel táctil. En esta tarjeta se mostrará el programa una vez ejecutado. Sólo cuenta con un conector de 40 pines para la conexión mediante cable plano a la PCB_1.

2. Requisitos software.

Para el programado del PIC y la posterior ejecución del programa se requiere del siguiente software.

2.1. MPLAB X.

Se requiere el entorno de desarrollo MPLAB X en cualquiera de sus versiones. También se podrá utilizar las versiones previas de MPLAB siempre que sean compatibles con los compiladores C32.

2.2. Compilador y librerías gráficas.

Se deberá añadir al entorno de desarrollo MPLAB un compilador compatible con PICs de 32 bits como el C32 o XC32.

También se instalarán en la ruta del MPLAB las librerías gráficas de Microchip.

2.3. MCP2515DM-BM PC Software Rev 2.0.

Para poder monitorear las tramas enviadas y recibidas en el CAN Bus instalaremos el software asociado a través del archivo ejecutable “BusMonitorSetup.exe” disponible en el CD del proyecto.

3. CONFIGURACIÓN DEL SISTEMA COMPLETO.

Los pasos recomendados para la configuración del sistema completo son los siguientes:

1. Comprobación previa de los voltajes de alimentación:

   Se desconectarán los 3 jumpers de PCB_1. A continuación se alimentará PCB_1 con una tensión de entre 12Vy 13.5V. Con un voltímetro se medirá JP6 y se comprobará que la tensión es de 6V. En caso de no serlo, se regulará mediante el potenciómetro.

   A continuación se conectará el jumper de 6V y se medirán las tensiones a la salida de los reguladores en los puntos de test 5V y 3.3V. Si las medidas son correctas se conectarán los dos jumpers restantes.

2. Se desconectará la alimentación de PCB_1.

3. Conexión de PCB_1 y PCB_2 mediante el conector de cable plano.

4. Abrir el proyecto en MPLABX y el software simulador de CAN Bus.

5. Conexión del programador ICD3 o PICKIT3 a PCB_1 y al PC mediante USB. El MPLABX deberá reconocer el programador.

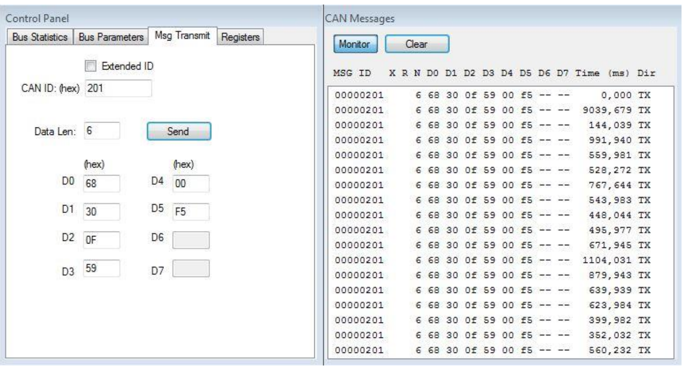
6. Conexión de una de las tarjetas simuladoras a PCB_1 y al PC mediante USB.

7. Configuración del software simulador.

   Para que el simulador de CAN reconozca la tarjeta pulsaremos en Device/Connect. Una vez reconocida la tarjeta ya tendremos acceso total al programa.

Previamente a que nos permita reconocer tramas enviadas en el bus y a enviarlas pasaremos a la pestaña Bus Parameters. Veremos que en Driver Mode estará activado el modo de configuración. Dejaremos todo como viene por defecto o cambiaremos la velocidad en el caso de utilizar otra (en nuestro caso 125kbit/s). Pulsaremos en Apply y ya tendremos el CAN de la tarjeta configurado.

Una vez configurado los parámetros del bus y desde la misma pestaña seleccionaremos Driver Mode en Normal. Si todo está correctamente configurado, a partir de este momento cualquier trama que pase por el Bus aparecerán en la parte derecha bajo CAN Messages. Una prueba sencilla para verificar que está correctamente configurado es enviar cualquier trama desde la pestaña Msg Transmit.



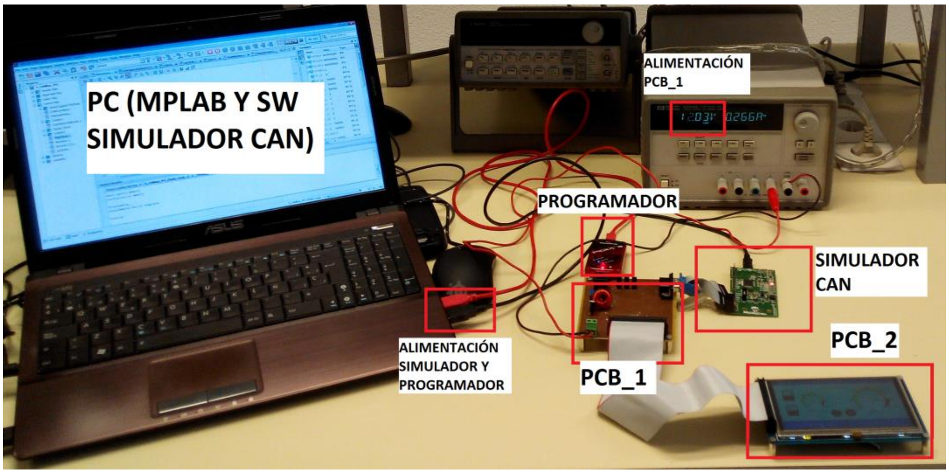
☐ Extended ID
CAN ID: (hex) 201
Data Len: 6

	(hex)		(hex)
D0	68	D4	00
D1	30	D5	F5
D2	0F	D6	
D3	59	D7	

```
MSG ID    X R N D0 D1 D2 D3 D4 D5 D6 D7 Time (ms) Dir
00000201      6 68 30 0f 59 00 f5 -- --     0,000 TX
00000201      6 68 30 0f 59 00 f5 -- --  9039,679 TX
00000201      6 68 30 0f 59 00 f5 -- --   144,039 TX
00000201      6 68 30 0f 59 00 f5 -- --   991,940 TX
00000201      6 68 30 0f 59 00 f5 -- --   559,981 TX
00000201      6 68 30 0f 59 00 f5 -- --   528,272 TX
00000201      6 68 30 0f 59 00 f5 -- --   767,644 TX
00000201      6 68 30 0f 59 00 f5 -- --   543,983 TX
00000201      6 68 30 0f 59 00 f5 -- --   448,044 TX
00000201      6 68 30 0f 59 00 f5 -- --   495,977 TX
00000201      6 68 30 0f 59 00 f5 -- --   671,945 TX
00000201      6 68 30 0f 59 00 f5 -- --  1104,031 TX
00000201      6 68 30 0f 59 00 f5 -- --   879,943 TX
00000201      6 68 30 0f 59 00 f5 -- --   639,939 TX
00000201      6 68 30 0f 59 00 f5 -- --   623,984 TX
00000201      6 68 30 0f 59 00 f5 -- --   399,982 TX
00000201      6 68 30 0f 59 00 f5 -- --   352,032 TX
00000201      6 68 30 0f 59 00 f5 -- --   560,232 TX
```

8. Alimentación de PCB_1. (12V-13.5V)

9. Pulsar en el botón “Make and Program Device” del MPLABX. A continuación pulsar “Debug Proyect” y la aplicación se mostrará en PCB_2.


PC (MPLAB Y SW SIMULADOR CAN)
ALIMENTACIÓN PCB_1
PROGRAMADOR
SIMULADOR CAN
PCB_2
ALIMENTACIÓN SIMULADOR Y PROGRAMADOR
PCB_1

ANEXO 3

CÓDIGO DEL PROGRAMA

Debido a la gran cantidad de código de que consta este proyecto, se ha decidido no incluirlo en este anexo. El código completo del programa se puede encontrar en en el CD adjunto al proyecto.